PYTHON WORKBOOK

LEARN HOW TO QUICKLY AND EFFECTIVELY PROGRAM WITH EXERCISES, PROJECTS, AND SOLUTIONS

PROGRAMMING LANGUAGES ACADEMY

Python Workbook

Learn How to Quickly and Effectively Program with Exercises, Projects, and Solutions

PROGRAMMING LANGUAGES ACADEMY

Table of Contents

Introduction

This workbook has been created for the purpose of practice and to allow readers of the crash course to further enhance their knowledge, understanding and usage of Python as a programming language.

If you have already gone through “Python Programming For Beginners: The Ultimate Beginner’s Guide to Learning the Basics of Python in a Great Crash Course Full of Notions, Tips and Tricks,” then this book is designed to further assist you in practicing all that you have learned so far.

Keeping true to the nature of the previous book, we will be looking into fun, intuitive and challenging exercises. These will test your ability and knowledge as a programmer and ensure that you are always prepared to tackle situations, questions and are able to identify errors. The exercises compiled here are taken from various sources, the links to which will be provided at the end of the book for your convenience. Should you like, you can visit these links for further exercises and test your programming skills to the max.

A great programmer is one who constantly practices his coding skills, has the ability to solve technical issues and can identify the kind of solution required to resolve a situation. Our aim with this book is in-line with this concept, which is why you can expect various types of exercises, projects and tests to learn from.

For your convenience, all solutions to questions and problems are provided in the last chapter. Refer to these when you feel you are unable to figure the problem out yourself. You may also refer to the first book from time to time to refresh your concepts and further clarify any ambiguities you may have during the process of learning and applying Python as a programming language.

There is no harm in admitting defeat. I assure you, I have been there a thousand times. If an exercise or question seems to be too much, remember to take it a little bit at a time. The better your state of mind, the clearer things will be for you.

Why Do I Need Exercises and Projects?

Ask any successful programmer in the world, and they will confirm the same: practice does make a programmer perfect. In the beginning, you might have been struggling with using PyCharm to write your code. Eventually, with a bit of practice, your pace started to increase. This is because you've now adapted to the IDE and the overall environment and feel of Python. The more you hone your skills, the more fluently you will type out your programs.

These exercises are designed to ensure you always have something to practice on. Once done with practicing the exercises as shown here, modify them at will to further create complex programs on your own. It is a perfect way to move into the intermediate and advanced levels of being a programmer. Every programmer goes through thousands of such miniature projects to gain perfect command of any language. While Python is comparatively easier to understand, do not underestimate the language. It can get tricky rather quickly, and with poor knowledge, you might actually end up going in circles.

The previous book has indeed taught you quite a few things. I have ensured that I created quizzes, exercises and questions to test each and every bit of that knowledge. This way, not only do you get to recall what you learned, but you also get to see the code in action by applying the solutions you think would be right. I do not claim to be the greatest programmer, nor that I am anywhere near being one, but what I can guarantee is that these exercises will certainly keep you on your toes and get you a step closer to becoming a great programmer yourself.

The projects are designed for your independent exploration. You have the entire internet to help you out with inspiration and ideas. Use the knowledge you gathered in the previous book and the experience that you will gain here to come up with more complex, more interactive programs that you can write for the projects I have chosen for you. These are projects which can make their way into the market as well, if you think out of the box and apply slightly more advanced logic and knowledge. Nothing is impossible, and programming is no different.

How Much Time Should You Spare?

While there are no specific limits that I would like to set here, I would, however, recommend that you squeeze out about one hour a day, if not more, to practice these exercises. Some of these might be quite easy in the beginning, but the purpose of such questions or exercises isn't to test your know-how alone; it is to push you further to come up with practical usage of the knowledge.

Let me also clarify that by one hour a day, I do not mean that you spend 60 minutes of your time pacing through these exercises alone. Copy these exercises to your IDE and work with them. By the end of the book, you should have enough exposure to Python that you will be able to develop your own code by first analyzing the situation, roughly noting down the logic that would help you and then writing the actual code itself.

Once you gain the habit of writing codes daily, you will progress at a phenomenal rate and will hopefully be well on your way to becoming a full-fledged programmer within no time.

With that said, it is time for us to begin the second phase of the journey. This is where we find out just how much you have learned thus far and whether you have picked up the concepts correctly to solve some complex issues and answer questions based on the technicalities of Python.

Here's a tip: Keep your PyCharm open as you read through the book to carry out the exercises as we move along. Do not jump to the last chapter just because the problem is seemingly impossible to solve. Take your time and analyze the situation carefully. The answer is far more obvious than you might think!

Chapter 1: Warm-Up Time

Well, first of all, my heartiest congratulations to you for picking up this book. If you have already completed the previous book, where I explained Python programming, double the wishes for yourself. The journey to Python is indeed one that is riddled with lines and lines of codes, waiting to be explored, understood and executed correctly.

We have gone through various chapters in the previous book and discovered so much about Python, starting from its history, all the way to the modern-day interpretation of automation, artificial intelligence and how things so advanced use Python.

We went through individual aspects of the language, such as the syntax, the variables, the data types, loops and functions, to count a few. All that is good, but the problem is we still don't know if we are ready to take on more advanced courses and learn things far beyond the scope of the previous book.

Fortunately, I already gave this some thought, which is why I will be providing you with various methods to fine-tune your skills further I am eager to get started, are you? Then let us dive into the exercises straight away and find out just where we stand.

Solutions to all exercises and questions are within the last chapter. Only consult them when you have tried everything possible to come up with the answer and somehow failed to do so.

Sea of Questions!

These questions, while they may sound easy, are designed to revisit some basic elements of python. Some of these may have options, while others may not. Do not be intimidated by these questions. Try and answer as many of these as possible.

Q-1: From the options given below, identify which of these is written in

Python?

Code-1:

```
using system;
var username = console.readline("Please enter your name: ");
console.write("Hello " + username);
```

Code-2:

```
<html>
    <head>
        <title>"Why write Python?"</title>
    </head>
    <body>
        <p>
            print("Hello World!")
        </p>
    </body>
</html>
```

Code-3:

```
import turtle
def my_function(name):
    print(f"Hello {name}")
my_function("Sam")
```

Code-4:

```
var name = "Mr. Marvel";
console.log("My name is " + name);
```

I am sure you see some familiar things here. Take your time and analyze each of these closely. The answer is right there, all it needs is a keen eye to pick it out! It's quite interesting to see that most of these seem to be using a familiar setup. It does take time for one to be fully familiar with the syntax, but once you are familiar, you should have no trouble figuring this out.

Visit any of the old exercises we did in the previous book. Try and match the way the code was written to the ones presented here. You should have your answer shortly!

Moving on to our next question, I promise it will be more questions and less of me, but I need to ensure I provide some explanations were needed to help those who may have picked up the book after a while, just to refresh their concepts.

Let us now begin a series of questions to test your understanding.

Q-2: How can you check if you have Python 3.8.x installed on your system?

A. Check if you have PyCharm installed on your system. If so, you have the latest Python version installed.

B. Run the command *python --version* in PyCharm to check for the version.

C. Run the command *python* in the command prompt for Windows to check the version. Run *python -v* on Mac and *python3* on Linux to get the version.

D. Visit the Python website to see if it can identify your version of Python.

Q-3: What is the language named after?

A. Monty Python's flying circus

B. The reptile Python

C. To honor an endangered species of Python

D. Just a random name that caught on

Q-4: How does each line in Python end?

A. With a colon ‘:’
B. With a semicolon ‘;’
C. With a full stop/period ‘.’
D. None of the above
E.

Q-5: What does the acronym IDE stand for?

A. International Day for Electronics
B. Integrated Developing Environment
C. Integrated Developer Environment
D. Integrated Developing Engineering

Q-6: How is a string represented in Python?

A. With a single quotation mark ‘’
B. With a double quotation mark “”
C. With either of the above.
D. None of the above.

Q-7: What is a variable?

A. It’s a function in Python.
B. It’s a method in Python.
C. It’s a user-created container holding immutable values.
D. It’s a user-created container holding values that can be modified.

Q-8: How would you print a string that says He said, “Yes!”?

A. print(“He said, “Yes!”)

B. print(f“He said, “Yes!”)

C. print(‘He said, “Yes!”’)

D. print(He said, Yes!)

Q-9: Running the code as shown, what will the output be?

```
num = '5' * '5'
print(num)
```

A. 25

B. 5, 5, 5, 5, 5

C. ‘5’ * ‘5’

D. TypeError: Can’t multiply sequence by non-int of type ‘str’

Q-10: If you run a code that ends up with an error, it will cause PyCharm to crash.

A. True

B. False

C. Depends on the type of code written

D. None of the above

Q-11: Which is the correct method to set the value for a bool ‘is_married’?

A. “True”

B. True

C. ‘True’

D. true

Q-12: Which of the following is a formatted string?

Code-1:

```
name = "Jiovanni"
age = 41
print("Hi, I am name and I am age years old")
```

Code-2:

```
name = "Jiovanni"
age = 41
print(f"Hi, I am {name} and I am {age} years old")
```

Code-3:

```
name = "Jiovanni"
age = 41
print("Hi, I am " + name + " and I am " + age + " years old")
```

Code-4:

```
name = "Jiovanni"
age = 41
print("Hi, I am [name] and I am [age] years old")
```

Q-13: To name a variable called first name, which method is correct? You can choose more than one answer to the following question.

A. firstname
B. FirstName
C. first.name
D. first_name

Q-14: Choose one or more answers which apply. Clean-code practice is:

A. To keep our workstations clean.

B. To name our variables and functions appropriately.

C. To improve readability.

D. To ensure the code is not breaking any laws.

Q-15: Which of the following is the correct way to create a variable named ‘test’:

A. def test():

B. test.create

C. test = “”

D. print(test)

Q-16: What is a concatenation of strings?

A. To merge two or more strings into a new string object

B. To separate two strings

C. To convert an integer into a string

D. None of the above

Q-17: Choose the correct answer(s). Python is:

A. The successor of ABC language

B. Only operable through PyCharm IDE

C. Used for automation and Machine Learning

D. All of the above

Q-18: What is OOP?

A. Object-Oriented Python

B. Object-Oriented PyCharm

C. Object-Oriented Programming

D. Only On Python

Q-19: How can you acquire user input and store it in a variable called income, for calculation purposes?

A. income = bool(“Enter your income:”)

B. income = int(“Enter your income:”)

C. income = input(“Enter your income:”)

D. income = int(input(“Enter your income:”)

Q-20: Which of these is/are true regarding Python?

A. There are two data types.

B. Variables can be called, modified or removed.

C. Python can work without PyCharm.

D. Python is a case-sensitive language.

That was quite a little warm-up, wasn’t it? We have only just begun. You may wish to check how many questions you got right by referring to the last chapter and seeing where you stand. These 20 questions were random, and most of them did not involve many technicalities.

Let us reflect on where you stand as a beginner:

If you achieved:

20 – Bravo! You answered each one of these brilliantly. That goes to show you were paying close attention to the book and the questions above. This is exactly the kind of result you should expect if you know your basics.

15 – 19 – So close to perfection, but do not let that bring you down. You did a fabulous job at answering these questions. Just revisit the ones you got wrong, and you should soon be polishing your skills to make it through with the finest results.

10 – 14 – There is room to improve. You did answer some of these questions rather well, but you have missed out on a few critical topics. It is best to

revisit the book and focus solely on the ones you got wrong. Go back to the concepts you missed and try to understand how the knowledge was applied here.

Below 10 – I have no other way to put it besides saying you need to work on your skills. It is quite likely that you were distracted during the test or when you were reading the book. Do not be disappointed though, failure is a part of the learning curve. As long as you have the will to learn, and a passion for pursuing, you will soon understand the issues and be writing your programs like anyone else.

I cannot stress enough that it is only through practice that you will be able to become the kind of programmer you aim to be one fine day. Just going through the book and answering every question right, only to stop practicing right after, will not serve you with any purpose or advantage.

There are a few things you should always remember as a programmer:

- Programming languages are constantly being updated. If you are out of action for a little while, you will soon be holding on to outdated knowledge.
- You are only as good as your coding skills. Anyone who is better will replace you instantly.
- Accuracy and speed of coding will differentiate between a good programmer and a struggling programmer, even if the latter knows everything.

Be sure to check out Python's official website for any updates that may arrive in the future. Currently, these codes were developed using Python 3.8.0, and they might have already been updated by the time you read this book. Always stay current with the latest version of Python and PyCharm.

Now that we have had a little stretch, it is time to dive into some technicalities and immerse ourselves deeper into the world of Python.

Is This Correct? - Part 1

This section will test out your skills in understanding the scenario and

identifying whether the code will work or if it needs to be modified. I will be presenting you with various scenarios in each chapter. They will grow in complexity and length, which is why it is a must that you read through each of the lines properly. Try not to copy and paste the code before you have come to a conclusion about the program's ability to work or fail. Let your brain to do the work first. Train yourself and your mind to think, analyze and resolve issues efficiently and effectively. Relying too much on the IDE will never allow you to explore your potential talent and problem-solving skills truly

Q-1: The program shown below was created to display a concatenated string. Do you think that the following will work? Will this deliver the required output of "This will be added With this!" or would it produce completely different output? If so, why?

```
string1 = "This will be added "

string2 = "With this!"

print("string1 + string2")
```

Q-2: The following is a program, written by a skilled programmer for a local business called "Pete's Garage" which should provide the business with a more reliable way of dealing with things. The program is written as shown below:

```
print('Welcome to Pete's garage)

name = input('Please enter your name: ')

job_number = int(input('Please enter your job number: '))

repair_cost = 100

discount = 15

total = repair_cost - discount

print("{name}, the total for {job_number} is ${total}")

print('Thank you for your business')
```

Will this program work? If not, can you identify the error that might cause this program to crash or stop responding?

Q-3: A university student decided to piece together a program that will allow potential online students from abroad to fill out the form and seek out further information regarding courses they're interested in. The form looks like this:

```
#Online Registration Form

print("Welcome to ABC Uni!")

print("Please enter the required information to begin.")

s_f_n = input("Enter your name: ")

phone = int(input("Enter your phone number: ")

em = "Enter your email: "

crs = "Choose your course:
```

The student has asked you to review the program and find out if there are any issues that need addressing. Find out what is wrong with the code and correct the issues.

Q-4: A student has created a program for a login page at a library for the new batch of users that have just joined. After a brief introduction, all users are asked to create their username and password pairs. After the users have entered their passwords, each password is compared to ensure that it matches. The program looks like this:

```
username = input("Username: ")

password = input("Password: ")

print("You have entered the following:")

print(username.lower())

print({password.lower()})

print(password==password.lower)
```

What seems to be the problem here? Why do you think the code will not work? What can be done to ensure that the program starts functioning?

Q-5: A programmer, with intermediate experience and knowledge, was asked to type out a program that would print out Boolean values for every grade a student acquires. For grades from A to B, the prompt was to print out True, while others would be considered False. Have a look at this code and see if this will work:

```
grades = ["A", "A", "B"", "U", "F", "E", "D"]

for x in grades:

    if x == A or x == B:

        x = True

        if x == True:

          print("Pass")

    else:

        x = False

        print("Fail")
```

The student has claimed that the code worked exceptionally. It is now up to you to analyze the code without testing it first to deliver your first impressions.

Bonus: Try and modify the same code with your own values and come up with something even better to practice.

Q-6: A string named initial_message contains the following message:

"*Hi, I have just taken part in the course. I hope that I will be a programmer one day.***"**

Without the quotation marks, a programmer was asked to find out the length of the string. She used the following method to do so:

```
String.length(initial_message)
```

Is this the correct method to check the length of a string? If not, what is the correct method applicable here?

Once you are done with these exercises, I do believe you will be in a better position to understand and analyze your understanding of the basic data types, the syntax, and the way variables work. I did add a few situations which were not exactly dealing with data types and variables; however, expect such questions and problems to come up in the future as well. Now, you should cross-check your answers with the last chapter to find out how much were you able to get and if you solved them correctly.

For those who have managed to score more than 70%, good job! For those who are struggling, I would recommend going through the first book and revisiting the data types, variables and input methods to refresh what you may have forgotten. Once again, there is no shame if you were not able to grasp the concept the first time.

Programming does take time and sometimes, even the best programmers spend days trying to figure out what is causing their program to crash, just to find out they may have missed out a single quotation mark, a comma, or misspelled a variable or a function. This is genuinely the case and happens almost every day.

It is with the introduction of PyCharm that our lives have been made so much easier. Imagine having to write entire programs, as long as 1000 lines, on notepad, which does not even come with the capability to check the spelling, let alone identify errors for any language after typing.

Use the power of IntelliSense, a technology used within PyCharm, that allows us to complete our codes with the click of a button. While that is time-saving, I would still recommend that you type out the complete code. If you make a habit of using PyCharm's IntelliSense, you might find yourself struggling should you join a firm that relies on Microsoft's VS Code IDE. It is always best to write the codes completely and proof-read the same where possible before you decide to execute them.

Now, we shall move on to our second chapter. We have revised and revisited the most basic concepts and principles of Python within this chapter. It is time to take the next step and start practicing with user-inputs, storing values

and recalling them. We will also look into matching information, and lastly, we will come across the first project that I would recommend every reader to go through and complete on their own.

All projects within the book have no definitive solutions. Apply the knowledge that you have gained so far to add to these projects and make them more complex and interactive.

Chapter 2: Recording Information

Programmers and their programs cannot exist without information. When I say information, I mean everything that has to do with data that is used for input, calculations, predictions, decision-making and eventual output. Getting the right information but storing the same in the wrong variable or having it tied up with elements that simply do not belong to each other can cause confusion, and at times, massive problems.

As a programmer, we need to ensure we know the kind of information we need for a specific program to run successfully. We need to ensure and ascertain that the information is stored accordingly and only called upon, modified or utilized when the situation calls for it.

We have covered quite a number of examples in the previous book, but here, we will go through certain scenarios that will further test your skills and critical thinking. Read the codes thoroughly, and you should be able to solve these with ease.

Storing/Recalling Information

While this section will mostly talk about the aspects of information itself, we will also be ensuring that we maintain our practice of clean-code to help programmers better understand the program itself. Expect a few issues where the program might be right, but the naming could use some help. In real-life situations, you will often encounter such issues and hence would need to re-address the names to make the program more meaningful and improve the overall readability of the code. With that said, let us begin!

Task-1: A YouTube streamer decided to conduct a survey where users were asked to provide feedback on what they would like to watch in the next stream. Your job is to create a program that uses the following

information and prints out the result of what the user chose, along with a thank you message.

What shall I stream next?

a) **Days Gone**
b) **Resident Evil 2**
c) **Fortnite**
d) **Apex Legends**
e) **Death Stranding**
f) **Surprise Us!**

The ending message should be:

You have chosen (option). I appreciate your time and hope to see you in the next one!

The exercise is fairly simple. Most of these will require you to print out information and then store the user-input value. Design the program so that it is able to understand what ‘a’ or ‘b’ or any character that the user chooses is, and then print the same out in the end greetings. You do not have to worry about printing out the name in the end. Just the letter of the selection will do for now.

Remember, the case-sensitive nature of the program still haunts us. Make use of methods like *.lower()* to ensure it matches our requirements. I will be providing my version of the program in the last chapter as a solution. For now, keep thinking, keep coding.

Hint: While you can use an ‘if’ statement, I would recommend bypassing that for now. Try and think of more basic means to store and recall values. This can easily be done without the use of logic and if/else statements. We will revisit this exercise in the future to make it a little more complex and appealing at the same time.

Task-2: A dentist wishes to have a program created for his website where the customers will be presented with multiple services. The customer will choose the option and will be presented with a total for the

service that is payable by the customer. The services are given as shown below:

a) **Root Canal Therapy - $250**
b) **Oral Hygiene Check - $50**
c) **Emergency Injury Treatment - $100**
d) **Post-Procedure Check-up - $150**
e) **Routine Check-ups and Consultation - $75**

For advanced payments, customers get a 50% discount.

Design a program that provides the customer with all the necessary information and gives a total according to what the customer chooses.

Confused? Let me give you a hint. You cannot store two values within a single variable unless you intend to create a list here. We are not aiming for that. You can either create two separate variables, service_a and price_a, or you can simply use conditional statements to further enhance the program. I leave the decision up to you.

I would prefer to use the latter though. It is a lot easier and less messy in nature.

So far, we have seen two simple exercises. Now, you will have probably realized that our programs can grow increasingly big and lengthy in nature the more complex we try to make them. The more information you have with you, the more lines of code will be used up. In typical situations, you will encounter programs that span well over 400 lines on average, and these are simple scripts used by programmers for various purposes.

I honestly do not mean to scare or intimidate you at all, but here's a little fact for you to digest.

The Mac OS X is believed to be the largest program ever written. It contains well over… wait for it… 85 million lines of code!

If you were to print that out and lay the regular A-4 sized papers in a straight line, you would cover a considerably large distance.

However, we are not here to set a world record, at least not yet. Our motive is rather simpler and kinder to our fingers and mind.

These exercises will test, or have already tested, your knowledge of creating variables, storing user-input information, modifying them (perhaps) and recalling them for simple calculations as well. At the heart of every program lies good and solid information. Without this, programs will have no reason to function.

So now that we have catered to these two exercises, let us move on ahead and see what else can we do.

Task-3: A college campus has decided to create a program that will determine the eligibility of an applicant based on a few questions and conditions. The college in question has asked you to create a program to record the following pieces of information:

a) **First name**
b) **Last name**
c) **Age**
d) **Overall score on their latest test result (out of 600)**
e) **If seeking scholarship**

Based on the following conditions, the eligibility for admission and for the scholarship will be decided:

For Admission:

- **The student should have achieved at least a 60% overall score or above for admission.**

For Scholarship:

- **The student must have at least a test score of 80% in order to be eligible for the scholarship.**

Create a program with data from three different students who have acquired a 471, 354 and 502 accordingly. Print out their results based on the above conditions.

This exercise will be fairly long and will require you to use the information and conditional statements to execute it to perfection. It might be wise to point out that such projects are actually in place and a loosely similar module is being used for Canada's Competitive Ranking System (CRS) score

calculation for immigration purposes as well.

When solving complex programs and situations, anticipate where you should set variables and use them. If you miss out on key positions, your program will have information but it will not function or be utilized properly.

It is essential to point out that you will need to have a basic understanding of mathematical operations like multiplication, division and so on, as these greatly help you create better programs.

Python allows you to get as creative as you can. If you believe you are bound by restrictions, you will be surprised to learn that there aren't any. You can apply Python to practically everything to come up with programs that can tell you if you have the right ingredients for a recipe or the right amount to pay off your debts. You can use it to make a predictive program that can predict possible outcomes so that you may prepare for all of the eventualities and so on.

Major business organizations use such programs which may seem simple but have quite a lot going on in the background. As a programmer, it is our job to realize what needs to be done, how it will be executed and how the desired result can be achieved. Always use a blank paper to draw out the flow chart. There is no specific flowchart for you to follow, which means that you can come up with your ideas on your own and map them out. It greatly helps with the programming part of the entire exercise. Go ahead and try it for the exercise above (if you haven't already done the exercise), otherwise, use one for any of the exercises you will encounter later on.

Is This Correct? - Part 2

Once again, we will be diving into some programs which I have created and/or compiled from various sources. Your goal is to see and analyze, *without using PyCharm or any IDE*, if they could or would work.

Q-1: A programmer came up with a program that would find the highest number from a given set of numbers. The numbers provided were stored as a list in a list variable called 'number_data' and the program that he designed looked like this:

```
number_data = [323, 209, 5900, 31092, 3402, 39803, 78341, 79843740, 895, 6749, 2870984]

for number in number_data:

    if num < number:

      num = number

print(num)
```

Will the above code work? What's wrong with the code?

Do not worry about the loops and if statements, try and analyze the error here. Take a guess and then check your answer. You can then try it on PyCharm to see whether things work or crash.

Q-2: A freelance programmer was tasked with creating a simple program to determine the eligibility of a profile for an auto-loan. Based on some specific information and conditions, such as the candidate should be less than 45 years of age, must have a minimum of a certain number as income and should not have any criminal records, the program was to determine if the same person was eligible for a loan or not. The programmer wrote the following program:

```
print("Your doorway to auto-loan eligibility check!")

print("Please provide complete information for best results")

name = input("Please enter your full name: ")

age = int(input("Enter your age: "))

income = int(input("Please enter your income per month: "))

nature_of_job = input("Do you work full-time, part-time or as a freelancer?: ")

has_license = input("Do you have a valid license? [y/n]: ")

if has_license.lower() == "y":

    has_license = True
```

```
else:
    has_license = False
has_criminal_record = input("In the last 5 years, do you have any criminal records? [y/n]: ")
if age > 45 and income >= 8000 and has_license == True and has_criminal_record == False:
    print("You are eligible for a loan")
elif age < 45 and income >= 5000 and has_license == True and has_criminal_record == False:
    print("You are eligible to apply for a loan")
elif has_criminal_record:
    print("You are not eligible for a loan")
elif income < 5000:
    print("You are not eligible at this time")
else:
    print("Please be patient as one of our specialists will be in touch!")
```

Upon executing a sample, the result was as follows:

```
Your doorway to auto-loan eligibility check!
Please provide complete information for best results
Please enter your full name: John Smith
Enter your age: 38
Please enter your income per month: 8300
Do you work full-time, part-time or as a freelancer?: Full-time
Do you have a valid license? [y/n]: y
```

In the last 5 years, do you have any criminal records? [y/n]: n

You are not eligible for a loan

Process finished with exit code 0

Do you think the program executed correctly? If not, what do you think the issue is?

Sometimes, the solution is rather obvious. You should be able to recognize the error straight away if you can link the dots and see which value is being recalled and what value/status the variable is printing. I cannot give you any more hints than that. Try and think this one through. Once sorted, let us move on to another one.

Q-3: As a school project, every student was asked to come up with a program that is no longer than 10 lines and is able to do some basic mathematics to produce answers. The student came up with a simple program that asks the user to type in a number and will let them know if the number is even or odd. The program is as shown below:

```
print("Setting the ODDS, EVEN!")

num = input("Enter a number: ")

if (num % 2) = 0:

    print("{0} is Even")

else:

    print("{0} is Odd")
```

Do you think the program will work? What errors, if any, do you think, would cause problems for the student?

Simple games can sometimes be quite entertaining. While checking the code myself, I spent over an hour working on it for no reason. I modified the values, the conditions and had quite the time.

Here is one more situation, see if you can figure out the issue on your own.

Q-4: As a side project, a programmer decided to create a simple

program that can let users know if the year, mentioned by the user, is a leap year or not. The leap year is calculated by determining if the year is exactly divisible by the number '4' and in the case of a century year, like the year 2000, it must be exactly divisible by 400.

Using the above concept, the programmer wrote this code:

```
print("My Brilliant Little Leap Year Calculator!")

year = int(input("Please enter the year: "))

if (year / 4) == 0:
    if (year / 100) == 0:
        if (year / 400) == 0:
            print(f"{year} is a leap year")
        else:
            print(f"{year} is not a leap year")
    else:
        print(f"{year} is a leap year")
else:
    print(f"{year} is not a leap year")
```

When the code was run with the year 2020, the following was the response:

```
My Brilliant Little Leap Year Calculator!

Please enter the year: 2020

2020 is not a leap year

Process finished with exit code 0
```

Why do you think that is?

(Programiz: https://www.programiz.com/python-

programming/examples/leap-year)

The above is actually a brilliant idea. Although we are a little too late, you can come up with more interesting ways to use such inspirations and programs to come up with something more meaningful.

Programmers across the world have been keenly looking into ways to further simplify things we normally overlook. Think good and think hard about matters in life which require some work and can be improved. Come up with your own genuine ideas or try and improve programs that already exist out there.

The internet is full of such programs. You will find these on a great many platforms, forums and social websites. Just browse through these, check out the source code and see if you have the right to modify the code to come up with something vibrant and improved.

I believe it is time for us to look into something new, something fresh and something that is genuinely challenging.

Project - 1

Before I provide you with your first project, let me quickly shed some light on what you can expect from these projects.

Every project will be unique as each one of us will have different ideas about how to carry out the task and execute the same. The projects will be designed to provide you with seemingly simple tasks, only to find out that you may have to do a little more than just copying and pasting blocks of code from one file to another.

Use your coding knowledge from all sources as these will not be bound to individual chapters. Projects are where you will encounter all kinds of problems, situations and scenarios. In order to solve these, or finish them successfully, you will need to use various methods right from the beginning, all the way to the end of complex matters like functions, classes and modules.

I will be providing you with links from which you can download specific

modules, libraries or classes to further help make the process easier. You already know how to import them into your PyCharm using the “from x import y” or “import xyz” method. Try and make a simple-looking scenario complex and interesting. Continue on developing these projects with advanced knowledge that you will hopefully gain after this book. A program is never truly complete. Even the best programs and software continue to be updated with newer knowledge, modules and variations.

Keep on practicing and adding more to these projects. Who knows, you might actually end up with something far superior and more useful than just a message that says “Hello World” at the end.

Task:

Create a simple game of “Rock, Paper, Scissors” where the computer randomly generates value and asks the user to input their selection. The result should show whether the user wins or loses, or if it is a draw.

Requirements:

In order to complete this project, you will need to use the following:

Packages:

From random import randint – This will be your first line of code. Random comes pre-installed and allows you to force the computer to randomize the selection. This will help you in ensuring that every turn is unique and unpredictable.

There are quite a few ways you can complete this project. As a reference, I will share my solution for this project at the end of the book as well.

Please note that I wish to encourage you to explore the world of Python and use your own genuine approaches, communicate with the community and learn better ways to code. For this reason, I will not share the details on the projects moving forward. I will gladly share the answers to questions and solutions to other problems. The rest, I invite you to use your power of deduction and programming to learn better.

So far, we have gone through some exercises, questioned what was right and what was not. We even initiated our very first project, which is quite challenging in all fairness. However, everything hinges on how well you understand your basics. The better you know them, the easier it will be to move from a beginner to an intermediate programmer and eventually to a skilled programmer. If you are unsure about certain aspects, it is always a good habit to revisit the concepts and revise what you have learned.

Time to say goodbye to variables and storing values and move on to our friends, the statements and loops.

Chapter 3

Running Around in Circles - Literally!

The world of loops and conditional statements is one that involves quite a lot of thinking. These will test your analytical and critical thinking, your problem-solving ability and will put you in rather uneasy spots. The trick behind each one of these is to carefully understand how they work.

As always, I would personally recommend using a pen and a paper, or your choice of text editor on your computer, to first draw out the scenario using flow charts and diagrams. To give you an idea, here is one:

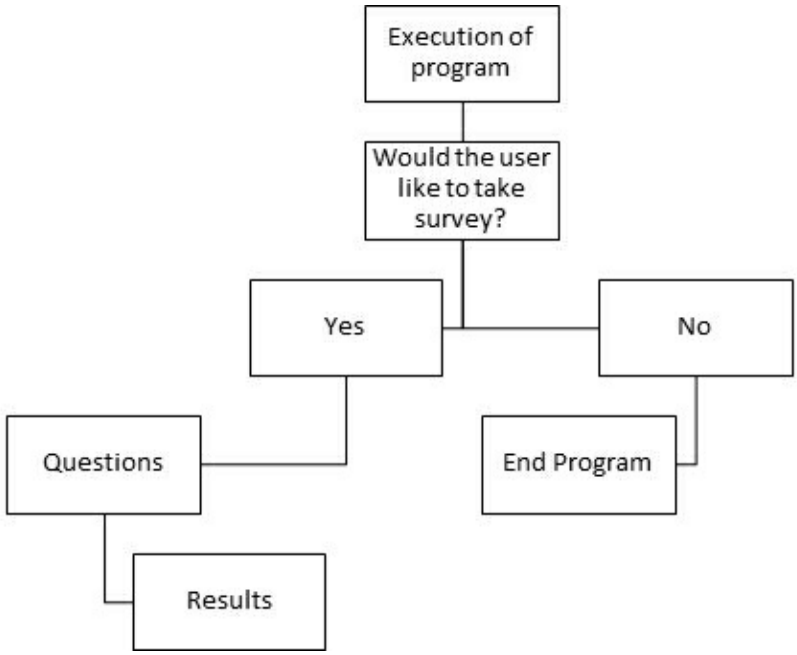

Table showing a simple 'if' conditional statement flow of a process.

Using tables or flow charts greatly helps us ease matters and develop a better understanding of the nature of the task at hand. Many great programmers first jot their ideas down into smaller parts to ensure that these are looked upon individually, where possible. By doing so, they can then focus on such charts to fully understand how the program needs to function and what the outcomes will be based on the path the user takes.

Similarly, it is highly recommended that you use such tables or charts to help

you fully develop a command of ‘if’ and ‘else’ statements and ultimately master them. There is no other way to say this so I will just say it: If you wish to create intelligent programs, you can never do so without developing a thorough understanding of these conditional statements and loops.

With that said, let us continue our quest to perfect our understanding of conditional statements and loops.

To ‘if’ or ‘for’ - That Is the Question!

Quite a lot of times, even I would spend days trying to figure out if we need to use the ‘if’ and ‘else’ conditions or opt to settle for a ‘for’ and ‘while’ combination. This is one of the trickiest aspects of programming, but it is one that can certainly deliver the program much-needed quality.

I will be posing you with various questions and scenarios. Where possible, I will also let you know of the desired outcome that you should achieve. Your job, as a programmer, is to figure out whether you need to use conditional statements, loops or a combination of both to achieve said outcome.

This will be tricky, which is why it is essential that you use your favorite IDE and have a go at these. Come up with your possible solutions and match those with the ones provided at the end of the book to see if you were successfully able to crack these open.

Task-1: A programmer has been asked to create a simple program where he is to map out digits from zero to nine in words. The program will ask a user to enter his/her number, and the program will print the same out in text instead. The desired result is as shown below:

Please, enter your number: ***415602397***

Output: ***Four One Five Six Zero Two Three Nine Seven***

How do you suppose this can be achieved? Would we need to use a loop here or a set of conditional statements?

Not as easy as it sounds, is it? I can start you off with a hint; use a dictionary

to create key-value pairs for numbers and words.

What you need to do next is absolutely your call. Take your time as there is no time limit imposed on you. Right now, you are practicing and learning how to be a better programmer. You can utilize your time well and without any deadlines to meet. In real-world scenarios, you may need to know all this beforehand so that you do not end up wasting time. If you are not able to do this, someone else will.

Here is our second task to think about and solve.

Task-2: A student carried out a program that calculated the shipping cost for an online retailer for the customer. The program would base the shipping cost on the total of the cart and the country of residence of the customer in question.

The chart below shows the shipping cost details:

Country	**Total**	**Shipping cost**
US	<$50 $50 - $99 $100 - $249 >$250	Free $10 $25 $50
AU	<$50 $50 - $99 $100 - $249 >$250	$10 $20 $50 $100
CA	<$50 $50 - $99	$5 $15

	$100 - $249 >$250	$30 $75
UK	<$50 $50 - $99 $100 - $249 >$250	$20 $25 $55 $110

Using the information above, the student was successfully able to create the program.

What do you think the student did?

This might actually be simple, but think this through and try to make this one as pleasing as possible. If you wish to take on a bit of a challenge, add lists and tuples to the mix to further test yourself.

Every program that you come across here can be done in hundreds of ways, if not thousands. To me, the glass may seem half empty and to you, it may seem half full. It is completely based on how we analyze things and look at them.

Put your thinking cap on and for the next one, do not copy and paste the code onto your PyCharm just yet. See if you can spot the issue with this one.

Task-3: You are a programmer who has been tasked with creating a simple yet intelligent game that stores a name that the users will have to guess. Upon providing the wrong name, the program will provide hints. You have created the following program, however, there seems to be something wrong here.

```
name = 'James'

guess = input("I have a name. Can you try to guess it?: ")

guess_num = 0

max_guess = 5

while guess != name and guess_num == max_guess:
```

```
    print(f"I am afraid, that's not quite right! Hint: letter {guess_num +1} ")
    print(guess_num + 1, "is", name[guess_num] + ". ")
    guess = input("Have another go: ")
    guess_num = guess_num + 1
if guess_num == max_guess and name != guess:
    print("Alas! You failed. The name was", name + ".")
else:
    print("Great, you got it in", guess_num + 1, "guesses!")
```

Try not to jump to your IDE to figure this one out. First, take a moment or two and analyze what is causing this program to end almost immediately upon providing an incorrect name. Surely, there must be something that is not right.

Read carefully between the code lines and you should soon be able to figure the matter out. Try and resolve the issue and then try the possible solution on your IDE to see if it works.

Once again, I do encourage you to make things better by modifying the code to your liking. There are hundreds of games you can come up with which are simple yet purely entertaining for others. As programmers, we take pride in knowing that we were able to execute codes with efficiency and ease. Programs will get more complex in nature as you progress along in your programming journey. For those interested in deep learning and machine learning, expect hundreds of lines of code to train the machine. To do that, you will need to be well-versed with almost every method in existence, all the functions and modules that are available across the internet to make the most out of the experience.

Project - 2

Time for yet another project. Since we are discussing games, create a Python program that lets the user know their astrological sign from the given date of birth. The program may seem rather easy, but once you look into the smaller details, you will soon realize that this will require you to think a little out of the box.

For this project, I will not be providing hints nor a model to follow. You already have the knowledge, and you should be able to execute this one with ease and a bit of finesse as well. You do not need any special modules or packages to get this project done. All you need is a quick search on the internet to see which star sign starts when to get you going.

Through trial and error, you should be able to create a program that is able to work easily and exceptionally. Should you encounter issues, try and resolve them on your own instead of looking for a solution on the internet.

If such projects interest you, you can find many more by searching for "Python projects for beginners" and get started. The more projects you work on, the better you will learn. Keep an eye out for what is in demand these days and set your target to one day be able to carry out programming of a level that will get you paid handsomely.

Questions and Answers

I wish this was a game where we could see who leads with how many points and know who answered the questions correctly. However, we will not let that get in the way of learning.

Answer as many questions as you can. To make this interesting, time yourself and try to answer all the questions within 10 minutes. At the end of the 10-minute mark, stop and review your answers to see where you stand.

Q-1: What does the == operator do?

A. It assigns a value

B. It recalls and matches the value of variables before and after it

C. It lets Python know not to equate variables

D. None of the above

Q-2: What is wrong with the code below?

```
x = 20
y = 30
z = 40
if x > y:
print("Something's wrong here")
```

A. Since x is less than y, the program will crash and return an error

B. z is not called, hence the program will not function

C. The condition is not followed by an indentation

D. There is nothing wrong with the program

Q-3: What will the result of the following program be?

```
alpha = 'Bravo'
bravo = 'Charlie'
charlie = 'Alpha'
for char in alpha:
    if char != 'a':
        print(char)
```

A. Bravo

B. Charlie

C. Alpha

D. None of the above

Q-4: What is the difference between 100 / 30 and 100 // 30?

A. It is just a typing mistake.

B. Both will deliver the same results.

C. The / will show a float figure while the // will show an integer remainder.

D. The / will show an integer remainder while the // a float remainder.

Q-5: What will the code shown below print as a result if a car is traveling at 75 miles per hour?

```
car_speed = int(input("Enter Car's current speed: "))
acceleration = 20 #per second
top_speed = 100
time = 0 #in seconds
if car_speed == 0:
    time = top_speed // acceleration
    print(f"It should take {time} second(s) for the car to reach its top speed")
    #For a stationary vehicle
else:
    time = (top_speed - car_speed) // acceleration
    print(f"it would take {time} second(s) to hit max speed.")
    #For a vehicle in motion
```

A. 5 second(s)

B. 3 second(s)

C. 1 second(s)

D. The program will return an error

Q-6: When should you use a ‘for’ loop?

A. When we need one specific output
B. When we need to iterate over a range of elements
C. When we wish to set a certain condition to be either true or false
D. None of the above

Q-7: What does the following error indicate?

Traceback (most recent call last):

File "C:/Users/Programmer/PycharmProjects/PFB/PFB-2/Project-2.py", line 1, in <module>

car_speed = int(input("Enter Car's current speed: "))

ValueError: invalid literal for int() with base 10: 'abc'

Process finished with exit code 1

A. The program crashed due to an invalid value entry
B. The program crashed as ‘abc’ was not entered as a string
C. Used single quotes instead of double quotes
D. None of the above

Q-8: The following program uses the log10 module from the ‘math’ package. The program was designed to carry out a few arithmetic operations to test the values and functionality of the program. What seems to be wrong here?

```
from math import log10

a = input("Enter 1st value: ")

b = input("Enter 2nd value: ")
```

print(a, "+", b, "is", a + b)

print(a, "-", b, "is", a - b)

print(a, "", b, "is", a * b)*

print(a, "/", b, "is", a / b)

print(a, "%", b, "is", a % b)

print(f"The base 10 logarithm of {a} is {log10(a)}")

*print(a, "^", b, "is", a**b)*

A. The strings are not formatted properly
B. The input values are stored as strings instead of integers
C. The log10 function will not work within a formatted string
D. All of the above

Q-9: What does an ‘elif’ statement do that ‘else’ can’t?

A. Elif conditions are secondary conditions that are executed when the main condition is false
B. Elif statements do not require conditions whereas else statements do
C. Elif statements are exactly the same as else statements
D. None of the above

Q-10: What does the following produce as a result?

def high_number(numbers):

max = numbers[0]

for number in numbers:

if number < max:

max = number

```
    return max
list = [21, 200, 31, 1, 39]
print(high_number(list))
```

A. 39

B. 5

C. 200

D. 1

Q-11: It is necessary to use ‘if’ statements every time you use loops. Is this statement true or false?

And stop! Hopefully, you were able to do all of these within the time limit of 10 minutes. I do not expect that this was an easy task as 10 minutes is not exactly much, but then again, it was a challenge.

Well done for scoring as many as you did. The challenge wasn’t to see who could get the most answers correct. It was actually designed to ensure you continue practicing. Even if you were able to get one of these correct, using your knowledge and not just a wild guess, it shows that you were paying attention and trying to understand the problem to come up with a solution. That is exactly the kind of attitude and commitment which will take us closer to success.

Take a break, if you have been practicing for a while. It is said that a human mind needs a quick minute or two to relax after every 45 minutes. You have earned it. Once you are back, let us move ahead and try to analyze more programs and see where people have gone wrong.

Is This Correct? - Part 3

This is honestly becoming my favorite section already. We get to see many programs and get to point the errors out, and in the process, we get to learn so much more. In fact, we were even inspired by some of these programs to do something similar. Keep an eye out for programs that may seem interesting as

you can always modify the way they work. However, this isn't always allowed by the law.

It is best to ensure you first know your rights regarding copying or editing said code. For this book, you do not have to worry about it much.

Q-1: A programmer decided to create a simple program, just to practice basic 'if' and 'else' conditions. He wrote the following program:

```
name = "John"

age = 33

is_married = True

is_happy = input("Are you happy?: ")

if is_happy.lower() == "yes":

    print("Well done!")

else:

    print('Sorry to hear that')
```

While the program runs fine, there is something that is wrong. Can you figure out what it is? You should be able to remove it and the program will still continue to function properly.

Q-2: A student defined a function as shown below:

```
def kms_to_miles(distance):

    distance * 0.621
```

When trying to use it, the program returned a value of 'None' as a result. Why do you think that happened?

A. The student must have not passed the appropriate parameter.

B. The distance used would have been in miles, hence the error.

C. The student forgot to use 'return' before the calculation while defining the function.

D. I have no idea why this failed. It should have worked.

Q-3: Do you think the following program should work? If not, why?

```
prices = [5, 10, 15, 20, 25]

total = 0

for item in prices:

    total += item

print(f"Your total price is: ${total}")
```

Do not try and copy the code to your IDE. Try and analyze the situation first to see if you can spot an error.

Q-4: In our previous book, we went through an example program as shown:

```
for an in range(3):

    for b in range(3):

        for c in range(3):

            print(f"({a}, {b}, {c})")
```

If you were to change the values of the ranges from top to bottom to 3, 2, 1, respectively, would the program work? What will the outcome be?

Q-5: According to a student, indentation is unnecessary and should not cause any problems when executing a program. The other student is of the idea that Python pays attention to whitespace and hence the indentation is quite important to maintain the code and arrange it accordingly. Which of the two do you think is right?

Q-6: Look at the code snippet below. It was taken from a program that was designed to iterate over key pairs of a dictionary.

```
output = ""

for char in number:
```

output += words.get(char) + " "

print(output)

What does the += operator do here?

Q-7: A teenager wanted to print out a simple design on python as a result. The design shown below was the output:

*

**

***

****

*****

******

****

****

****

****

I did this!

****

****

****

****

******

*****

****

***

**

*

Do you think this can be done using loops? If so, can you code the program?

For those who may remember the days when DOS was considered a new thing, we surely created a lot of these patterns and designs. Now, Python can carry that out but instead of just relying on print statements, we use loops to do extensive work for us instead.

There are more modules and packages such as turtle, which helps you create some incredible designs. Turtle is a built-in, pre-installed package, and you should be able to import it easily. You can easily find tutorials online to see what the entire thing is all about, and possibly use the same to see just how effortlessly Python can do our work for us.

Python is far more than just a coding language. You can get so much done by just typing a few lines. With the help of your favorite IDE, things are sure to enhance the experience and keep the learning curve high. With python, you always get to learn something new.

Speaking of new, it is time for us to look into our next chapter and come across another aspect of programming which we covered in the last book: Functions!

Chapter 4

Using the Right Functions

We have already seen how functions can greatly help us with carrying out and organizing massive blocks of code into a simple, word or two long recallable functions. If that isn't enough, Python brings forth some of the finest pre-defined functions to the table to further assist us in carrying out so many other tasks flawlessly.

While we will not be diving into functions that are far above the scope of a beginner, we will still be looking into problems where we will see whether given functions would help us out or not.

Take some time to ensure you go through the previous book in case you need to refresh your memory and revisit the concepts behind what functions are and how they operate. Once sorted, let us proceed with our exercises, questions and scenarios.

Getting Functioning Programs to Work

The objective of this section of the book is to reinforce the concepts related to functions. To make this a little more interesting than others, I will be providing you with scenarios to carry out on your own and convert into fully functioning programs. You will be provided with a solution at the end. All you need is to ensure you read the scenario and visualize it to sort out what needs to go where, and then form a fully functional program.

From this point forward, I will be providing you with various scenarios, each consisting of programs that you will need to create. Some of these may require you to do a bit of research as well. However, if any special method, library, module or package is used, I will provide those within the instructions.

Q-1: You are to create a function that calculates a taxi fare. The taxi fare is comprised of a base fare of $3.00 and then $0.10 for every 100 meters traveled. Create a function that takes distance as its only parameter (in km) and returns the value of the total applicable fare. Follow up with a program to show the functioning nature of the function.

The situation is rather easy. All you need is to work on the function and work out the details accordingly. The rest will just fall into place automatically. Take your time and process the information.

I know, it is easy to be overwhelmed. A quick look at the internet will only leave you confused. Take small steps and start somewhere. With time and a bit of understanding, you will soon grasp the concept and be able to figure out such situations easily.

The next one to come is even tougher. We have entered the phase where we can say goodbye to the five to 10 line programs. It is best to let them go and practice on bigger, more complex programs to polish your skills and make yourself step out of your comfort zone and properly start exploring.

Q-2: A client has asked you to create a reusable program containing reusable functions. The first situation is to create a function that creates a virtual deck of playing cards.

Since there are 52 cards, the cards with numeric values from two to nine will be represented by their respective numbers. For 10, Jack, Queen, King and Ace, you are required to use T, J, Q, K and A.

Following the numeric/alphabetic value will be another character to represent the suit. Use h for hearts, c for clubs, d for diamonds and s for spades.

You should create a function that does not take any parameter and uses loops to iterate through all the cards and store them with a two-character abbreviation in a list. The function should only return this list as a result.

Hint: You will need to use the following as your beginning line.

from random import randrange

For this, expect quite a few lines of code. Your complete knowledge about Python will be tested and tried in this one and the ones to come ahead. There is no need to rush into things. If you are not able to do it the first time around, you can do a bit of research and get some suggestions. I would recommend not to jump to the solutions right away. Try and push your brain to think a little outside the box. There is much to be learned, and the only way it can be learned is to try.

Q-3: You have been asked by a colleague to help out with a Python project. The assignment is to create a function that can generate a random password for a user which would contain between 6 to 8 characters at most.

For this, you will need to use the following:

from random import randint

shortest_pass = 6

max_pass = 8

min_ASCII = 33

max_ASCII = 126

The function should generate a random value from positions 33 to 126 of the ASCII table. This function will not take in any parameters.

The above exercise will test you further and might even require a little research on the ASCII table, if you haven't seen one before. It is always nice to carry out a bit of research as it greatly helps us as a programmer to further excel at what we do.

One of the toughest calls to make when creating a function is knowing the right parameter you need to use to make the function work. You can use almost any argument you pass through the parentheses to make it into a parameter. Sometimes, you may not need a parameter at all, as we saw in some of the exercises above. In such cases, it is best to leave the parameter blank.

When in doubt, always consult the Python documentation to learn more about

the various parameters you can make use of when defining functions. For recalling functions, hover your mouse over the parentheses and a little prompt should display the kind of parameters you can use with said function.

Is This Correct? - Part 4

Q-1: Below is a user-made function that is designed to iterate through a given range and look for the highest number. Will the function work when it is called?

```
def high_number(numbers):
        max = numbers[0]
        for number in numbers:
          if number < max:
            max = number
        return max
list = [21, 200, 31, 1, 39]
high_number(list)
```

Q-2: What seems to be the issue with the following?

```
def this_function():
    print("Hello From This Function!")
this_function_with_args(name, greeting):
    print(f"Hello {name}, From This Function!, I wish you {greeting}")
this_function()
this_function_with_args()
```

Q-3: What would this function do?

```
def plus(a,b):
```

```
  sum = a + b
(sum, a)
sum, a = plus(3,4)
print(sum)
```

Q-4: Can you place a loop within a function as shown below?

```
def plus(*args):
  total = 0
  for i in args:
      total += i
  return total
print(plus(20,30,40,50))
```

Now that we have revisited quite a few concepts and methods, and functions as well, let us head over to our final project and see what it is all about.

Final Project – Hangman

Remember the old game called Hangman? The one that involved blanks and a limited number of guesses to guess a movie, a name, a person, a city or something else? For your final project, I decided to come up with a tough one. A project that will use almost everything you have learned.

To make it even better, if you execute the program correctly, you can store it as a recallable function or create a separate package so that you can use this over and over again. While there are hundreds of variations for this game online, use your own unique approach. Feel free to browse the internet to get some inspiration.

Requirements:

You will need to use the following as your first line:

```
import time
```

Let this be a project that you are proud of, once it is finished. By the end of this project, rest assured you are ready to take on the challenges and offer some exquisite programming skills to those who require programmers such as yourself.

The journey of a programmer does not end here. There are far too many things that lay ahead which you will need to keep pace with. Learn about various libraries, modules and packages to see how they can bring refinement to your projects.

For those interested in Machine Learning, Automation, Artificial intelligence and Deep Learning, you will come across some names like Scikit-Learn, Turtle and a few more. While it is still too early to jump into these, it is a good idea to have a look at them and see how they perform in action.

Chapter 5

The Solutions!

Finally, a chapter where you do not have to worry about solving exercises. You have gone through every curveball, every scenario and every exercise provided within this workbook, and for that, you deserve the heartiest congratulations. I am glad that you took your best shot at solving these exercises.

For every programmer, the beginning is always the biggest hurdle. Once you set your mind to things and start creating a program, things automatically start aligning. The needless information is automatically omitted by your brain through its cognitive powers and understanding of the subject matter. All that remains then is a grey area that we discover further through various trials and errors.

There is no shortcut to learn to program in a way that will let you type codes 100% correctly, without a hint of an error, at any given time. Errors and exceptions appear even for the best programmers on earth. There is no programmer that I know of personally who is able to write programs without running into errors. These errors may be as simple as forgetting to close quotation marks, misplacing a comma, passing the wrong value and so on. Expect yourself to be accompanied by these errors and try to learn how to avoid them in the long run. It takes practice but there is a good chance you will end up being a programmer who runs into these issues only rarely.

With that said, it is time to shift our focus back to the main part of this chapter, the solutions to the never-ending exercises. Some of these were incredibly simple while others were not as simple as they sounded. Regardless of how many you get right, you should never be afraid of nor let down by failure. It is a part of our learning cycle and hence should be accepted, understood and then corrected. The more you learn from your mistakes, the easier things will be in the future.

Chapter by Chapter Solutions

Chapter 1 Solutions

Below are the answers and the correct answer applicable to the question is highlighted in bold text.

Q-1: The question asked to identify which of the codes mentioned used Python as its programming language.

- Code 1: Unity C# - We do not use the statement ‘using’ in Python, nor do we use semicolons at the end.
- Code 2: HTML – The second language in the list used HTML, which is used for developing websites.
- **Code 3: Python – Evident from the import statement and the missing semicolon.**
- Code 4: JavaScript – Similar to C# but with a few differences.

Q-2: How can you check if your system has Python 3.8.x installed on your system? Let’s see what the correct method is to check for the same:

A. PyCharm – PyCharm is an IDE and can be installed without having Python on board.
B. Command on PyCharm – The given command will not work on PyCharm.
C. Command on console/terminal – This is the correct method to locate the version of python.
D. The website will not provide an indication of the version you have installed.

Q-3: What is the language named after? This was related to the history that we covered in the previous book. If you got this wrong, it is okay. The correct answer is A. The language was named after the famous Monty Python’s flying circus.

Q-4: How does each line in Python end? To be honest, this was one of the easiest questions. But surprisingly, people still tend to forget.

A. With a colon ‘:’
B. With a semicolon ‘;’
C. With a full-stop/period ‘.’

D. **None of the above – Because every line in Python ends without any special character except when defining conditions or functions.**

Q-5: What does the acronym IDE stand for? We have discussed this in the previous book.

A. International Day for Electronics
B. Integrated Developing Environment
C. Integrated Developer Environment
D. Integrated Developing Engineering

Q-6: How is a string represented in Python?

A. With a single quotation mark ‘’
B. With a double quotation mark “”
C. With either of the above. – We can create strings using single or double quotation marks.
D. None of the above.

Q-7: What is a variable? Once again, an easy question.

A. It’s a function in Python.
B. It’s a method in Python.
C. It’s a user-created container holding immutable values – close but not exactly the right answer as variables can change.
D. It’s a user-created container holding values that can be modified.

Q-8: How would you print a string that says He said, “Yes!”?

A. print(“He said, “Yes!”)
B. print(f“He said, “Yes!”)
C. print(‘He said, “Yes!”’) – Using double quotation marks would end the string at the start of “Yes”
D. print(He said, Yes!)

Q-9: Running the code as shown, what will the output be?

```
num = '5' * '5'

print(num)
```

A. 25
B. 5, 5, 5, 5, 5
C. ‘5’ * ‘5’
D. TypeError: Can’t multiply sequence by non-int of type ‘str’ – A string cannot be multiplied by another string.

Q-10: If you run a code that ends up with an error, it will cause PyCharm to crash.

A. True
B. False – PyCharm is a safe working environment that is designed to test codes. If the code will crash, the IDE returns the kind of error code to inform users that this will crash when used outside PyCharm.
C. Depends on the type of code written
D. None of the above

Q-11: Which is the correct method to set the value for a bool ‘is_married’?

A. “True”
B. True
C. ‘True’
D. true

Q-12: Which of the following is a formatted string?

Code-1:

```
name = "Jiovanni"

age = 41

print("Hi, I am naming and I am age years old")
```

Code-2:

name = "Jiovanni"

age = 41

print(f"Hi, I am {name} and I am {age} years old")

This is the correct way to format a string. There are other ways you can format strings, but I personally find this a lot easier.

Code-3:

name = "Jiovanni"

age = 41

print("Hi, I am " + name + " and I am " + age + " years old")

Code-4:

name = "Jiovanni"

age = 41

print("Hi, I am [name] and I am [age] years old")

Q-13: To name a variable called the first name, which method is correct? You can choose more than one answer to the following question.

- **A. firstname**
- **B. FirstName**
- C. first.name
- **D. first_name**

If you are surprised about the second entry here, remember that it is recommended not to use this format to name variables. While it is not recommended, it does not mean that it is wrong.

Q-14: Choose one or more answers which apply. Clean-code practice is:

- A. To keep our workstations clean.
- **B. To name our variables and functions appropriately**
- **C. To improve readability.**

D. To ensure the code is not breaking any laws.

Q-15: Which of the following is the correct way to create a variable named 'test':

A. def test():
B. test.create
C. test = "" – You can always create variables as blanks
D. print(test)

Q-16: What is a concatenation of strings?

A. To merge two or more strings into a new string object
B. To separate two strings
C. To convert an integer into a string
D. None of the above

Q-17: Choose the correct answer(s). Python is:

A. The successor of ABC language – This was discussed in the previous book.
B. Only operable through PyCharm IDE
C. Used for automation and Machine Learning – These are two of the most in-demand fields of modern times.
D. All of the above

Q-18: What is OOP?

A. Object-Oriented Python
B. Object-Oriented PyCharm
C. Object-Oriented Programming – Remember, everything in Python is considered an object.
D. Only On Python

Q-19: How can you acquire user input and store it in a variable called income, for calculation purposes?

A. income = bool("Enter your income:")
B. income = int("Enter your income:")

C. income = input(“Enter your income:”)
D. income = int(input(“Enter your income:”) – The user will input a value that will be then converted into an integer.

Q-20: Which of these is/are true regarding Python?

A. There are two data types
B. Variables can be called, modified or removed.
C. Python can work without PyCharm
D. Python is a case-sensitive language

Yes! Python can work without PyCharm as PyCharm is just one of the many ways to operate Python and write codes.

Is This Correct? – Part 1 Solutions

This was indeed a section that caused quite a bit of confusion but provided the perfect opportunity to practice our coding skills and keep an eye out for errors. Let us look at the solutions to the questions posed within the first part of the “Is This Correct?” series.

Q-1: The program shown below was created to display a concatenated string. Do you think that the following will work? Will this deliver the required output of “This will be added With this!” or would it produce completely different output? If so, why?

string1 = "This will be added "

string2 = "With this!"

print("string1 + string2")

Ans: The actual answer would be as shown below if you were to run this code as it is:

string1+string2

This happened because the print statement was given a string that contained the characters string1 and string2. Python took them quite literally as a string

object. In order to make them work properly, you will need to use a formatted string. The solution would be as shown here:

string1 = "This will be added "

string2 = "With this!"

print(f"{string1}{string2}")

Output:

This will be added With this!

Q-2: Following is a program, written by a skilled programmer for a local business called “Pete’s Garage” which should provide the business with a more reliable way of dealing with things. The program is written as shown below:

print('Welcome to Pete's garage)

name = input('Please enter your name: ')

job_number = int(input('Please enter your job number: '))

repair_cost = 100

discount = 15

total = repair_cost - discount

print("{name}, the total for {job_number} is ${total}")

print('Thank you for your business')

Will this program work? If not, can you identify the error that might cause this program to crash or stop responding?

Ans: To begin with, there are a few errors that need addressing. Beginning at the very top, the first print statement is not correctly written. The string ends at the apostrophe after Pete and hence the program will not function nor identify the rest.

Moving forward, the print statement declaring the total is not formatted. It is missing the ‘f’ key character. Once these two errors are sorted, the program

should function rather well.

If you try the program now, it should display this:

Welcome to Pete's garage

Please enter your name: Emma

Please enter your job number: 91829

Emma, the total for 91829 is $85

Thank you for your business

Process finished with exit code 0

Q-3: A university student decided to piece together a program that will allow potential online students from abroad to fill out the form and seek out further information regarding courses they're interested in. The form looks like this:

#Online Registration Form

print("Welcome to ABC Uni!")

print("Please enter the required information to begin.")

s_f_n = input("Enter your name: ")

phone = int(input("Enter your phone number: ")

em = "Enter your email: "

crs = "Choose your course:

The student has asked to review the program and find out if there are any issues that need addressing. Find out what is wrong with the code and correct the issues.

Ans: Once again, we see that the phone variable has two opening parentheses but only one closing parenthesis. Fix that first to ensure that this line of code is functional. Next, the 'em' variable has a fixed string value. But we are trying to get an input from the user. Use the input function here. Lastly, the 'crs' has an opening quotation mark but no closing quotation mark. It is also

missing the input function.

Lastly, if needed, a print statement can be added to confirm the selection. The overall program should look like this:

```
#Online Registration Form

print("Welcome to ABC Uni!")

print("Please enter the required information to begin.")

s_f_n = input("Enter your name: ")

phone = int(input("Enter your phone number: "))

em = input("Enter your email: ")

crs = input("Choose your course: ")

print(f"{s_f_n}, you have chosen {crs} as your course.")

print(f"Details of the {crs} course will be emailed to you at {em}")

print(f"We may also be in touch via a phone call at {phone}")
```

Output:

Welcome to ABC Uni!

Please enter the required information to begin.

Enter your name: Joel

Enter your phone number: 915789654

Enter your email: joel@abcxyz.com

Choose your course: Game Development

Joel, you have chosen Game Development as your course.

Details of the Game Development course will be emailed to you at joel@abcxyz.com

We may also be in touch via a phone call at 915789654

Q-4: A student has created a program for a login page at a library for the new batch of users that have just joined. After a brief introduction, all users are asked to create their username and password pairs. After the users have entered their passwords, each password is compared to ensure that it matches. The program looks like this:

```
username = input("Username: ")

password = input("Password: ")

print("You have entered the following:")

print(username.lower())

print({password.lower()})

print(password==password.lower)
```

What seems to be the problem here? Why do you think the code will not work? What can be done to ensure that the program starts functioning?

Ans: The program would not have worked as it would have printed out a case-sensitive password in all lower-case. A comparison between the two would then obviously return 'false' as they would not be the same.

Remember, Python is a case-sensitive language. Password is not the same as PASSWORD or password. To make this a fully functioning program, you will need to do a little more than just tweak parentheses or fill out missing values.

Here is my take on the same program.

```
username = input("Username: ")

password = input("Password: ")

print("You have entered the following:")

print(username)
```

```
print(password)
login = False
login_id = input("Please enter your username: ")
login_pwd = input("Enter your password: ")
while not login:
    if login_id == username and login_pwd == password:
        print("You have logged in successfully")
        login = True
    elif login_id == username and login_pwd != password:
        print("You have entered the wrong password")
        break
    elif login_id != username and login_pwd == password:
        print("You have entered the wrong ID!")
        break
    else:
        print("You have entered an invalid ID/Password")
        break
```

Q-5: A programmer, with intermediate experience and knowledge, was asked to type out a program that would print out Boolean values for every grade a student acquires. For grades from A to B, the prompt was to print out True, while others would be considered False. Have a look at this code and see if this will work:

```
grades = ["A", "A", "B"", "U", "F", "E", "D"]
for x in grades:
    if x == A or x == B:
```

```
        x = True
        if x == True:
          print("Pass")
    else:
        x = False
        print("Fail")
```

The student has claimed that the code worked exceptionally. It is now up to you to analyze the code without testing it first to deliver your first impressions.

Ans: Once again, we see some misplaced quotation marks in the first line. The first order of the day is to correct those right away. Then we have our ‘if’ statements. See how the statement says if x == A. Here, we need to add quotation marks in order for A to be a string object instead of a separate variable. Here is the end result:

```
grades = ["A", "A", "B", "U", "F", "E", "D"]
for x in grades:
        if x == 'A' or x == 'B':
          x = True
          if x == True:
            print("Pass")
        else:
          x = False
          print("Fail")
```

Output:

Pass

Pass

Pass

Fail

Fail

Fail

Fail

Q-6: A string named initial_message contains the following message:

"*Hi, I have just taken part in the course. I hope that I will be a programmer one day.***"**

Excluding the quotation marks, a programmer was asked to find out the length of the string. She used the following method to do so:

String.length(initial_message)

Is this the correct method to check the length of a string? If not, what is the correct method applicable here?

Ans: No. That is not the correct way to check the length of the string. The correct way to do so is using the *len()* function, as shown here:

print(len(initial_message))

This will then count the characters and print out the value of characters in return.

Chapter 2 Solutions

Task-1: A YouTube streamer decided to conduct a survey where users were asked to provide feedback on what they would like to watch in the next stream. Your job is to create a program that uses the following information and prints out the result of what the user chose, along with a thank you message.

What shall I stream next?

a) **Days Gone**
b) **Resident Evil 2**
c) **Fortnite**
d) **Apex Legends**
e) **Death Stranding**
f) **Surprise Us!**

The ending message should be:

You have chosen (option). I appreciate your time and hope to see you in the next one!

Ans: This is indeed quite an easy one to do. Here is the solution for the survey that this streamer is trying to create.

print("Welcome to my survey, where I ensure I deliver what you want!")

print("Please, take some time to fill this out and help me decide what to play next!")

a = "Days gone"

b = "Resident Evil 2"

c = "Fortnite"

d = "Apex Legends"

e = "Death Stranding"

f = "Surprise Us"

print(f"""Here are your options. Remember, select one:

a){a},

b){b},

c){c},

d){d},

e){e},

f){f}""")

selection = input("Please make a selection: ")

print(f"You have chosen {selection}. I appreciate your time and hope to see you in the next one!")

And the corresponding output would look like this:

Welcome to my survey, where I ensure I deliver what you want!

Please, take some time to fill this out and help me decide what to play next!

Here are your options. Remember, select one:

a)Days gone,

b)Resident Evil 2,

c)Fortnite,

d)Apex Legends,

e)Death Stranding,

f)Surprise Us

Please make a selection: b

You have chosen b. I appreciate your time and hope to see you in the next one!

And that would be as simple as that. If you were able to do this easily, well done. If not, now you have every idea of how it works. Do remember that this is a very basic level of programming. You can make this as interactive as you wish. Now, you can try using various methods to make it even better.

Task-2: A dentist wishes to have a program created for his website where the customers will be presented with multiple services. The customer will choose the option and will be presented with a total for the service that is payable by the customer. The services are given as shown below:

a) **Root Canal Therapy - $250**
b) **Oral Hygiene Check - $50**
c) **Emergency Injury Treatment - $100**
d) **Post-Procedure Check-up - $150**
e) **Routine Check-ups and Consultation - $75**

For advanced payments, customers get a 50% discount.

Design a program that provides the customer with all the necessary information and gives a total according to what the customer chooses.

Ans: I already gave you a hint, but now, let us see how this would work.

```
print("The Patient's Portal")

print("Please select the service you would like to come in for.")

a = "Root Canal Therapy"

print(f"A){a}")

b = "Oral Hygiene Check"

print(f"B){b}")

c = "Emergency Injury Treatment"

print(f"C){c}")

d = "Post-procedure checkup"

print(f"D){d}")

e = "Routine Checkups and consultation"

print(f"E){e}")

selection = input("Please choose one: ")

print(f"You chose {selection}")

total = 0

if selection.lower() == "a":
```

```
        total = 250
        print(f"Your total is ${total}")
elif selection.lower() == "b":
        total = 50
        print(f"Your total is ${total}")
elif selection.lower() == "c":
        total = 100
        print(f"Your total is ${total}")
elif selection.lower() == "d":
        total = 150
        print(f"Your total is ${total}")
else:
        total = 75
        print(f"Your total is ${total}")
print("Did you know? Book in advance and get 50% off!")
payment_time = input("Would you like to pay today? [y/n]: ")
if payment_time.lower() == "y":
        total = total * 0.50
        print(f"Your total payable is ${total}")
else:
        print(f"You will need to pay ${total} at the counter.")
print("Have a smiling day!")
```

The output should show you the following:

The Patient's Portal

Please select the service you would like to come in for.

A)Root Canal Therapy

B)Oral Hygiene Check

C)Emergency Injury Treatment

D)Post-procedure checkup

E)Routine Checkups and consultation

Please choose one: a

You chose a

Your total is $250

Did you know? Book in advance and get 50% off!

Would you like to pay today? [y/n]: y

Your total payable is $125.0

Have a smiling day!

The program is a little long, but it does keep you engaged and keeps you on your toes to type in the correct code in order to make a successful program. If you were able to figure this one out on your own, brilliant. If not, it is perfectly okay. There is plenty of time to learn what you need to do in order to carry out various tasks to come up with your desired results.

Task-3: A college campus has decided to create a program that will determine the eligibility of an applicant based on a few questions and conditions. The college in question has asked you to create a program to record the following pieces of information:

a) **First name**
b) **Last name**
c) **Age**
d) **Overall marks in their latest test result (out of 600)**

e) **If seeking scholarship**

Based on the following conditions, the eligibility for admission and for the scholarship will be decided:

For Admission:

- **The student should have achieved at least a 60% overall score or above for admission.**

For Scholarship:

- **The student must have at least a test score of 80% in order to be eligible for the scholarship.**

Create a program with data from three different students where they have acquired 471, 354 and 502 accordingly. Print out their results based on the above conditions.

Ans: This program also required some critical thinking and posed a bit of a challenge with three different situations. To address these, we need our friendly 'if' statements to help us out and create an intelligent program.

```
print("Welcome to the Applicant's eligibility checker")

first_name = input("Please enter your first name: ")

last_name = input("Please enter your last name: ")

age = input("Please enter your age: ")

marks = float(input("Please enter your overall marks (out of 600): "))

total_marks = 600.0

passing_marks = total_marks * 0.60

marks_for_scholarship = total_marks * 0.80

if marks >= passing_marks:

    print(f"Congratulations {last_name}! You are eligible for admission to the college!")
```

```
    scholarship = input("Are you seeking a scholarship? [Y/N]: ")
    if scholarship.lower() == "y" and marks >= marks_for_scholarship:
      print(f"Congratulations {last_name}! You are eligible for a scholarship!")
    elif scholarship.lower() != "y":
      pass
    else:
      print(f"{last_name}, you are not eligible for a scholarship at this time!")
else:
    print(f"Unfortunately {last_name}, you are not eligible for admission.")
print("Thank you for your input and we wish you good luck!")
```

The output for the above would be as shown below:

```
Welcome to the Applicant's eligibility checker
Please enter your first name: John
Please enter your last name: Doe
Please enter your age: 21
Please enter your overall marks (out of 600): 402
Congratulations Doe! You are eligible for admission to the college!
Are you seeking a scholarship? [Y/N]: y
Doe, you are not eligible for a scholarship at this time!
Thank you for your input and we wish you good luck!
```

Is This Correct? – Part 2 Solutions

Q-1: A programmer came up with a program that would find the highest

number from a given set of numbers. The numbers provided were stored as a list in a list variable called 'number_data' and the program that he designed looked like this:

```
number_data = [323, 209, 5900, 31092, 3402, 39803, 78341, 79843740, 895, 6749, 2870984]

for number in number_data:

    if num < number:

      num = number

print(num)
```

Will the above code work? What's wrong with the code?

Ans: The above code will not work as the variable 'num' is not defined. While some may have been unnecessarily worried, the solution was rather simple. Just declare a variable called num with a starting value of zero.

```
number_data = [323, 209, 5900, 31092, 3402, 39803, 78341, 79843740, 895, 6749, 2870984]

num = 0

for number in number_data:

    if num < number:

      num = number

print(num)
```

Remember to declare variables before they are used as Python reads the program line by line.

Q-2: A freelance programmer was tasked with creating a simple program to determine the eligibility of a profile for an auto-loan. Based on some specific information and conditions, such as the candidate should be less than 45 years of age, must have a minimum of a certain amount as income and should not have any criminal records, the program was to determine if the same person was eligible for a loan or

not. The programmer wrote the following program:

```
print("Your doorway to auto-loan eligibility check!")

print("Please provide complete information for best results")

name = input("Please enter your full name: ")

age = int(input("Enter your age: "))

income = int(input("Please enter your income per month: "))

nature_of_job = input("Do you work full-time, part-time or as a freelancer?: ")

has_license = input("Do you have a valid license? [y/n]: ")

if has_license.lower() == "y":

    has_license = True

else:

    has_license = False

has_criminal_record = input("In the last 5 years, do you have any criminal records? [y/n]: ")

if age > 45 and income >= 8000 and has_license == True and has_criminal_record == False:

    print("You are eligible for a loan")

elif age < 45 and income >= 5000 and has_license == True and has_criminal_record == False:

    print("You are eligible to apply for a loan")

elif has_criminal_record:

    print("You are not eligible for a loan")

elif income < 5000:

    print("You are not eligible at this time")
```

else:

print("Please be patient as one of our specialists will be in touch!")

Upon executing a sample, the result was as follows:

Your doorway to auto-loan eligibility check!

Please provide complete information for best results

Please enter your full name: John Smith

Enter your age: 38

Please enter your income per month: 8300

Do you work full-time, part-time or as a freelancer?: Full-time

Do you have a valid license? [y/n]: y

In the last 5 years, do you have any criminal records? [y/n]: n

You are not eligible for a loan

Process finished with exit code 0

Do you think the program executed correctly? If not, what do you think the issue is?

Ans: The above program has one major issue which will cause the program to always push out not eligible as a result. The has_criminal_record has not yet been defined as True or False. For that, we need to create a separate condition. Here is the fully operable program with a sample test.

print("Your doorway to auto-loan eligibility check!")

print("Please provide complete information for best results")

name = input("Please enter your full name: ")

age = int(input("Enter your age: "))

income = int(input("Please enter your income per month: "))

nature_of_job = input("Do you work full-time, part-time or as a freelancer?:

```
")
has_license = input("Do you have a valid license? [y/n]: ")
if has_license.lower() == "y":
    has_license = True
else:
    has_license = False
has_criminal_record = input("In the last 5 years, do you have any criminal records? [y/n]: ")
if has_criminal_record.lower() != "y":
    has_criminal_record = False
else:
    has_criminal_record = True
if age > 45 and income >= 8000 and has_license == True and has_criminal_record == False:
    print("You are eligible for a loan")
elif age < 45 and income >= 5000 and has_license == True and has_criminal_record == False:
    print("You are eligible to apply for a loan")
elif has_criminal_record:
    print("You are not eligible for a loan")
elif income < 5000:
    print("You are not eligible at this time")
else:
    print("Please be patient as one of our specialists will be in touch!")
```

Output:

Your doorway to auto-loan eligibility check!

Please provide complete information for best results

Please enter your full name: Elliot Charington

Enter your age: 41

Please enter your income per month: 6900

Do you work full-time, part-time or as a freelancer?: full

Do you have a valid license? [y/n]: y

In the last 5 years, do you have any criminal records? [y/n]: n

You are eligible to apply for a loan

Q-3: As a school project, every student was asked to come up with a program that is no longer than 10 lines and is able to do some basic mathematics to produce answers. The student came up with a simple program that asks the user to type in a number and will let them know if the number is even or odd. The program is as shown below:

```
print("Setting the ODDS, EVEN!")

num = input("Enter a number: ")

if (num % 2) = 0:

    print("{0} is Even")

else:

    print("{0} is Odd")
```

Do you think the program will work? What errors do you think, if any, would cause problems for the student?

Ans: This program has a few issues. Firstly, notice that the print statements are not formatted. Secondly, the condition set is not using the right comparison operator. Instead of '=' we need to use the '==' operator to

provide the condition with a comparison point.

Moreover, the input will initially be stored as a string. Be sure to use the *int* converter. Lastly, replace the ‘0’ in the print statements with the variable ‘num’ and that should do it. If you run the program now, it should be fully functional and should return the correct results.

```
print("Setting the ODDS, EVEN!")

num = int(input("Enter a number: "))

if (num % 2) == 0:

        print(f"{num} is Even")

else:

        print(f"{num} is Odd")
```

Q-4: As a side project, a programmer decided to create a simple program that can let users know if the year, mentioned by the user, is a leap year or not. The leap year is calculated by determining if the year is exactly divisible by the number ‘4’ and in the case of a century year, like the year 2000, it must be exactly divisible by 400.

Using the above concept, the programmer wrote this code:

```
print("My Brilliant Little Leap Year Calculator!")

year = int(input("Please enter the year: "))

if (year / 4) == 0:

   if (year / 100) == 0:

        if (year / 400) == 0:

          print(f"{year} is a leap year")

        else:

          print(f"{year} is not a leap year")

   else:
```

```
        print(f"{year} is a leap year")
else:
    print(f"{year} is not a leap year")
```

When the code was run with the year 2020, the following was the response:

My Brilliant Little Leap Year Calculator!

Please enter the year: 2020

2020 is not a leap year

Process finished with exit code 0

Why do you think that is?

Ans: This one involved a little more mathematics than actual programming. Using the '/' operator, we would not have received the exact values. Instead, we will need to change all the '/' operators to '%' and run the program again. Now, the year 2020 will be shown as a leap year, and any other leap years will be correctly calculated as well.

```
print("My Brilliant Little Leap Year Calculator!")
year = int(input("Please enter the year: "))
if (year % 4) == 0:
    if (year % 100) == 0:
        if (year % 400) == 0:
          print(f"{year} is a leap year")
        else:
          print(f"{year} is not a leap year")
    else:
        print(f"{year} is a leap year")
```

```
else:
    print(f"{year} is not a leap year")
```

Output:

My Brilliant Little Leap Year Calculator!

Please enter the year: 2020

2020 is a leap year

Project – 1 Solution

(Exercise 118: Stephenson, B. (2014) The Python Workbook)

As promised, here is how I have created a simple game of “Rock, Paper, Scissor” using Python.

```
from random import randint
t = ["Rock", "Paper", "Scissors"]
computer = t[randint(0, 2)]
tries = 0
player = " "
while tries <= 9:
        player = input("Rock, Paper, Scissors?")
        tries += 1
        if player == computer:
          print("Tie!")
        elif player == "Rock":
          if computer == "Paper":
          print("You lose!", computer, "covers", player)
```

```
  else:
    print("You win!", player, "smashes", computer)
elif player == "Paper":
  if computer == "Scissors":
    print("You lose!", computer, "cut", player)
  else:
    print("You win!", player, "covers", computer)
elif player == "Scissors":
  if computer == "Rock":
    print("You lose...", computer, "smashes", player)
  else:
    print("You win!", player, "cut", computer)
else:
  print("That's not a valid play. Check your spelling!")
computer = t[randint(0, 2)]
```

You may copy this and try it out yourself. Change the values where possible to see how it affects the program. This is one terrific way to spend some quality time with a game while learning the dynamics as well.

Chapter 3 Solutions

Task-1: A programmer has been asked to create a simple program where he is to map out digits from zero to nine in words. The program will ask a user to enter his/her number, and the program will print the same out in text instead. The desired result is as shown below:

Please, enter your number: ***415602397***

Output: ***Four One Five Six Zero Two Three Nine Seven***

How do you suppose this can be achieved? Would we need to use a loop here or a set of conditional statements?

Ans: To achieve this simple goal, we will use a dictionary to create key-value pairs. Here is the solution, and honestly it is worth a try.

```
number = input("Phone number: ")

words = {
    "1": "One",
    "2": "Two",
    "3": "Three",
    "4": "Four",
    "5": "Five",
    "6": "Six",
    "7": "Seven",
    "8": "Eight",
    "9": "Nine",
    "0": "Zero"
}

output = ""

for char in number:
    output += words.get(char) + " "

print(output)
```

This should now print out words instead of numbers.

Task-2: A student carried out a program that calculated the shipping

cost for an online retailer for the customer. The program would base the shipping cost on the total of the cart and the country of residence of the customer in question.

What do you think the student did?

Ans: Refer to the original chart in the question for reference purposes. A successful version of the program would be as shown here:

```
total = int(input("Please enter the total amount: "))
country = input("Country [US/AU/CA/UK]: ")
if country.upper() == "US":
    if total <= 99 and not total <= 49:
      print("Shipping Cost is  $10")
    elif total >= 100 and not total >= 250:
      print("Shipping Cost is $25")
    elif total >= 250:
      print("Shipping Costs $50")
    else:
      print("FREE")
if country.upper() == "AU":
    if total <= 99 and not total <= 49:
      print("Shipping Cost is  $20")
    elif total >= 100 and not total >= 250:
      print("Shipping Cost is $50")
    elif total >= 250:
      print("Shipping Costs $100")
```

```
        else:
          print("Shipping Cost is $10")
if country.upper() == "CA":
        if total <= 99 and not total <= 49:
          print("Shipping Cost is  $15")
        elif total >= 100 and not total >= 250:
          print("Shipping Cost is $30")
        elif total >= 250:
          print("Shipping Costs $75")
        else:
          print("Shipping Cost is $5")
if country.upper() == "UK":
        if total <= 99 and not total <= 49:
          print("Shipping Cost is  $25")
        elif total >= 100 and not total >= 250:
          print("Shipping Cost is $55")
        elif total >= 250:
          print("Shipping Costs $110")
        else:
          print("Shipping Cost is $20")
```

The output would provide you the correct answers according to the input.

Task-3: You are a programmer who has been tasked with creating a simple yet intelligent game that stores a name that the users will have to guess. Upon providing the wrong name, the program will provide hints.

You have created the following program, however, there seems to be something wrong here.

```
name = 'James'

guess = input("I have a name. Can you try to guess it?: ")

guess_num = 0

max_guess = 5

while guess != name and guess_num == max_guess:

        print(f"I am afraid, that's not quite right! Hint: letter {guess_num +1}
")

        print(guess_num + 1, "is", name[guess_num] + ". ")

        guess = input("Have another go: ")

        guess_num = guess_num + 1

if guess_num == max_guess and name != guess:

    print("Alas! You failed. The name was", name + ".")

else:

        print("Great, you got it in", guess_num + 1, "guesses!")
```

Ans: Believe it or not, there was only one slight error. The program worked, but the problem was that it would accept even an incorrect entry as correct. If you pay close attention to the ‘while’ condition, you will notice that guess_num == max_guess is never met, hence this block of code never gets executed. When that happens, there is nothing to add an increment to the number of guesses. The program would then move on to the ‘if’ statement, and hence choose the ‘else’ part as the output. To correct this, all you needed to do was to replace the ‘while’ condition to this:

```
while guess != name and guess_num != max_guess:
```

Now, the program should function properly and make this into an interactive little game.

Question and Answers

Q-1: What does the == operator do?

A. It assigns a value
B. It recalls and matches the value of variables before and after it
C. It lets Python know not to equate variables
D. None of the above

Q-2: What is wrong with the code below?

```
x = 20
y = 30
z = 40
if x > y:
print("Something's wrong here")
```

A. Since x is less than y, the program will crash and return an error
B. z is not called, hence the program will not function
C. The condition is not followed by an indentation
D. There is nothing wrong with the program

Q-3: What will the result of the following program be?

```
alpha = 'Bravo'
bravo = 'Charlie'
charlie = 'Alpha'
for char in alpha:
    if char != 'a':
     print(char)
```

A. Bravo
B. Charlie
C. Alpha

D. None of the above – The result would print “B r a v o”

Q-4: What is the difference between 100 / 30 and 100 // 30?

A. It is just a typing mistake
B. Both will deliver the same results
C. The / will show a float figure while the // will show an integer remainder
D. The / will show an integer remainder while the // a float remainder

Q-5: What will the code shown below print as a result if a car is traveling at 75 miles per hour?

```
car_speed = int(input("Enter Car's current speed: "))

acceleration = 20 #per second

top_speed = 100

time = 0 #in seconds

if car_speed == 0:

    time = top_speed // acceleration

    print(f"It should take {time} second(s) for the car to reach its top speed")

    #For a stationary vehicle

else:

    time = (top_speed - car_speed) // acceleration

    print(f"it would take {time} second(s) to hit max speed.")

    #For a vehicle in motion
```

A. 5 second(s)
B. 3 second(s)
C. 1 second(s) – We are using the // operator, which will always

return an integer value

D. Program will return an error

Q-6: When should you use a ‘for’ loop?

A. When we need one specific output
B. When we need to iterate over a range of elements
C. When we wish to set a certain condition to be either true or false
D. None of the above

Q-7: What does the following error indicate?

Traceback (most recent call last):

File "C:/Users/Programmer/PycharmProjects/PFB/PFB-2/Project-2.py", line 1, in <module>

car_speed = int(input("Enter Car's current speed: "))

ValueError: invalid literal for int() with base 10: 'abc'

Process finished with exit code 1

A. The program crashed due to an invalid value entry
B. The program crashed as ‘abc’ was not entered as a string
C. Used single quotes instead of double quotes
D. None of the above

Q-8: The following program uses the log10 module from the ‘math’ package. The program was designed to carry out a few arithmetic operations to test the values and functionality of the program. What seems to be wrong here?

```
from math import log10

a = input("Enter 1st value: ")

b = input("Enter 2nd value: ")

print(a, "+", b, "is", a + b)

print(a, "-", b, "is", a - b)
```

print(a, "", b, "is", a * b)*

print(a, "/", b, "is", a / b)

print(a, "%", b, "is", a % b)

print(f"The base 10 logarithm of {a} is {log10(a)}")

*print(a, "^", b, "is", a**b)*

A. The strings are not formatted properly
B. The input values are stored as strings instead of integers
C. The log10 function will not work within a formatted string
D. All of the above

Q-9: What does an ‘elif’ statement do that ‘else’ can’t?

A. Elif conditions are secondary conditions that are executed when the main condition is false.
B. Elif statements do not require conditions whereas else statements do.
C. Elif statements are exactly the same as else statements.
D. None of the above.

Q-10: What does the following produce as a result?

```
def high_number(numbers):
    max = numbers[0]
    for number in numbers:
      if number < max:
        max = number
    return max
list = [21, 200, 31, 1, 39]
print(high_number(list))
```

A. 39

B. 5
C. 200
D. 1 – If you chose 200 because I named the function high_number, you missed out on the fact that I used a < operator instead of >, hence the result would always be the smallest number in the list.

Q-11: It is necessary to use 'if' statements every time you use loops. Is this statement true or false?

Ans: False – There have been many cases where we did not use 'if' statements when we used a loop.

Is This Correct? – Part 3 Solutions

Q-1: A programmer decided to create a simple program, just to practice basic 'if' and 'else' conditions. He wrote the following program:

```
name = "John"

age = 33

is_married = True

is_happy = input("Are you happy?: ")

if is_happy.lower() == "yes":

    print("Well done!")

else:

        print('Sorry to hear that')
```

While the program runs fine, there is something that is wrong. Can you figure out what it is? You should be able to remove it and the program will still continue to function properly.

Ans: The is_married condition is neither needed nor being used anywhere. This is only making the program a little more confusing. It is best to remove anything from the program will never be used. The program should still be able to run properly.

Q-2: A student defined a function as shown below:

```
def kms_to_miles(distance):
    distance * 0.621
```

When trying to use it, the program returned a value of 'None' as a result. Why do you think that happened?

A. The student must have not passed the appropriate parameter.
B. The distance used would have been in miles, hence the error.
C. The student forgot to use 'return' before the calculation while defining the function.
D. I have no idea why this failed. It should have worked.

Q-3: Do you think the following program should work? If not, why?

```
prices = [5, 10, 15, 20, 25]
total = 0
for item in prices:
    total += item
print(f"Your total price is: ${total}")
```

Ans: The program is actually error-free. This program should run fine on its own.

Q-4: In our previous book, we went through an example program as shown:

```
for a in range(3):
    for b in range(3):
        for c in range(3):
            print(f"({a}, {b}, {c})")
```

If you were to change the values of the ranges from top to bottom to 3, 2, 1, respectively, would the program work? What will the outcome be?

Ans: The program should continue to work and the outcome will be the following:

(0, 0, 0)

(0, 1, 0)

(1, 0, 0)

(1, 1, 0)

(2, 0, 0)

(2, 1, 0)

Q-5: According to a student, the indentation is unnecessary and should not cause any problems when executing a program. The other student is of the idea that Python pays attention to whitespace and hence the indentation is quite important to maintain the code and arrange it accordingly. Which of the two do you think is right?

Ans: The latter has the correct answer as Python pays significant attention to whitespaces. This is exactly why Python will fail to function if you forget to use an indentation where it is necessary.

Q-6: Look at the code snippet below. It was taken from a program that was designed to iterate over key pairs of a dictionary.

```
output = ""
for char in number:
    output += words.get(char) + " "
print(output)
```

What does the += operator do here?

Ans: Although the program written above will fail to work, owing to multiple issues, the actual question is highlighting what the += operator would do. This operator adds as an increment the value of words.get(char) to the output on every loop.

Q-7: A teenager wanted to print out a simple design on python as a result. The design shown below was the output:

*

**

***

****

*****

******

****

****

****

****

I did this!

****

****

****

****

******

*****

****

***

**

*

Do you think this can be done using loops? If so, can you code the program?

Ans: There are two ways you can carry out this program. The first one involves a lot of lines, each using a print statement. That is far too basic and we will simply ignore that. The second one involves the use of loops, and that is where things get interesting. Here is how you can achieve this:

```
character = '*'

for char in range(0, 11):

    output = character * char

    print(output)

print("I did this!")

for char in range(10, 0, -1):

    output = character * char

    print(output)
```

You have first informed Python to iterate through the range in ascending order. In the second loop, you have used a -1 step in the end and reversed the order of the range. Now, it will start from 10 and end at the last number of the range.

Chapter 4 Solutions

Q-1: Taxi Fare Calculator

Ans: This was actually a little tough. There were quite a few components which needed to be readdressed a few times before arriving at the final phase. Here is my defined function to carry out the taxi fare calculations.

```
def calculate_fare():

    distance = float(input("Enter the distance in kms: "))

    distance = distance * 1000  #Convert kms into meters
```

```
    fare = 3.0 + ((distance / 100) * 0.1) # extra fare charged at every 100 meters
    return fare
print(f"Your total taxi fare is ${calculate_fare()}")
```

Remember, you will need to use a float value to get the correct answers.

Q-2: Card Deck Shuffler

Ans: This one was hard. I am sure you had to do a bit of research, and if you did, it is perfectly okay. I was not expecting you get this one right immediately. Using functions can sometimes be tricky, especially if it involves quite a few elements. Here is how this is done:

```
from random import randrange
def createDeck():
    cards = []
    for suit in ["s", "h", "d", "c"]:
      for value in ["2", "3", "4", "5", "6", "7", "8", "9", "T", "J", "Q", "K", "A"]:
        cards.append(value + suit)
    return cards
def shuffle(cards):
    for i in range(0, len(cards)):
      other_pos = randrange(0, len(cards))
      temp = cards[i]
      cards[i] = cards[other_pos]
      cards[other_pos] = temp
def main():
```

```
    cards = createDeck()
    print("This is the original deck: ")
    print(cards)
    print()
    shuffle(cards)
    print("This is the shuffled version: ")
  print(cards)
main()
```

Now, the result should show you both the original cards and the shuffled version as well.

Q-3: Random Password Generator

Ans: The solution to this is as shown below. Most of it is pretty much self-explanatory.

```
from random import randint
shortest_pass = 6
max_pass = 8
min_ASCII = 33
max_ASCII = 126
def randomPass():
    randomLength = randint(shortest_pass, max_pass)
    result = ""
    for i in range(randomLength):
      randomChar = chr(randint(min_ASCII, max_ASCII))
      result = result + randomChar
```

```
    return result
def main():
    print("Your randomly generated password: ", randomPass())
main()
```

I encourage you to try and change the values per your liking to change the outcome and see how the overall program is affected.

Is this correct? – Part 4 solutions

Q-1: Below is a user-made function that is designed to iterate through a given range and look for the highest number. Will the function work when it is called?

```
def high_number(numbers):
    max = numbers[0]
    for number in numbers:
      if number < max:
        max = number
    return max
list = [21, 200, 31, 1, 39]
high_number(list)
```

Ans: The above is actually a code you have already visited in this workbook. Currently, it will not print out anything as there is no print command used. In order to print the result, use the print function and place the function with the parameter within the parentheses of the print function. Before executing the code, be sure to change the < operator to > to view the highest number in the list.

Q-2: What seems to be the issue with the following?

```
def this_function():
```

```
    print("Hello From This Function!")

this_function_with_args(name, greeting):

    print(f"Hello {name}, From This Function!, I wish you {greeting}")

this_function()

this_function_with_args()
```

Ans: In the above code, the function “this_function_with_args()” is not defined as the key element of ‘def’ is missing. Without it, Python will not be able to find anything called this_function_with_args() and hence the program will fail to execute.

Once that is sorted, you will need to pass two arguments for this function to work properly. Here is an example:

```
this_function_with_args("John", "Happy Birthday")
```

Now, the function will work as per the requirements.

Q-3: What would this function do?

```
def plus(a,b):

  sum = a + b

(sum, a)

sum, a = plus(3,4)

print(sum)
```

Ans: This function will not work as it is not defined properly and is honestly a mess. Remember, our aim is to write clean code so that everyone can understand what is going on. Here is the solution to make this function work properly:

```
def plus(a, b):

        return a + b

sum = plus(3, 4)
```

print(sum)

Now, this function will calculate the two numbers that are passed as arguments.

Q-4: Can you place a loop within a function as shown below?

```
def plus(*args):
  total = 0
  for i in args:
      total += i
  return total
print(plus(20,30,40,50))
```

Ans: Yes, the function should work perfectly fine. When defining a function, you can write hundreds of lines using ‘if’ and ‘else’ conditions as well as ‘for’ and ‘while’ loops. You can also use constructors, other predefined functions, methods and values to make the function more meaningful. There is no limitation to how much code a function can hold.

Where to Head Next

The world is now your digital playground. You have all the essentials you need to get started. You are already fluent with your basics and should have no problem tackling the bigger challenges that lay ahead of you.

To make the most of the journey, utilize great resources such as udemy, coursera, YouTube and many other major names that offer more advanced learning opportunities. There are countless books out there that offer the same learning experience with great detail. Be sure not to miss out on programming practice as the more you wait, the more you lose track of the concepts you worked so hard to learn.

Continue making great and simple programs to keep yourself in touch with the basics. Remember, the stronger the basics, the easier it is for you to advance.

Work on various projects which you can find on many websites and freelance platforms. For those willing to seek a career, consider the option of working in Artificial Intelligence, Deep Learning, Machine Learning and other related fields. These are advancing every day and require more talented programmers such as yourself to go out there and make a difference.

Conclusion

We were excited when we began this workbook. Then came some arduously long tasks which quickly turned into irritating little chores that nagged us as programmers and made us think more than we normally would. There were times where some of us even felt like dropping the whole idea of being a programmer in the first place. But, each and every one of us who made it to this page, made it through with success.

Speaking of success, always know that your true success is never measured properly nor realized until you have hit a few failures along the road. It is a natural way of learning things. Every programmer, expert or beginner, is bound to make mistakes. The difference between a good programmer and a bad one is that the former would learn and develop the skills while the latter would just resort to Google and locate an answer.

If you have chosen to be a successful Python programmer, know that there will be some extremely trying times ahead. The life of a programmer is rarely socially active either unless your friend circle is made up of programmers only. You will struggle to manage your time at the start but once you get the hang of things, you will start to perform exceptionally well. Everything will then start aligning and you will begin to lead a more relaxed lifestyle as a programmer and as a human being.

Until that time comes, keep your spirits high and always be ready to encounter failures and mistakes. There is nothing to be ashamed of when going through such things. Instead, look back at your mistakes and learn from them to ensure they are not repeated in the future. You might actually be able to make programs even better or update the ones which are already functioning well enough.

Lastly, let me say it has been a pleasure to guide you through both these books and to be able to see you convert from a person who had no idea about Python to a programmer who now can code, understand and execute matters at will. Congratulations are in order. Here’s a digital cheers for you!

print(“Bravo, my friend!”)

I wish you the best of luck for your future and hope that one day, you will look back on this book and this experience as a life-changing event that led to

a superior success for you as a professional programmer. Do keep an eye out for updates and ensure you visit the forums and other Python communities to gain the finest learning experience and knowledge to serve you even better when stepping into the more advanced parts of Python.

References

Briggs, J, R. (2013): Python For Kids. San Francisco, CA: No Starch Press

Matthes, E. (2016): Python Crash Course. San Francisco, CA: No Starch Press

Payne, B. (2015): Teach Your Kids to Code. No Starch Press

Programiz. (n.d.). Python Program to Check Leap Year. Retrieved December 10, 2019, from https://www.programiz.com/python-programming/examples/leap-year.

Stephenson, B. (2014): The Python Workbook. Springer International Publishing