LEARN PYTHON

Peace Adebanji and Lawrence Orolobo

YEOLOM

Learn Python Book by Lawrence Orolobo Olisa (@lawrenceorolobo on Instagram)

This Book was reviewed by Anthonia Barasua Lawrence and PeaceAdebanji.

The book, along other valuable pdfs are available on Yeolom's MarketPlace (yeolom.com/marketplace).

INTRODUCTION

You might be thinking "why should I read this book?"
"Programming is hard; there is no point."
"I am involved in art not science, so I don't need programming."

Frankly, It's really not my place to dictate to you whether or not you should learn programming. I wish only to explain the very basics of programming in Python and considering the fact that you've read to this point, it must mean there is indeed a desire or interest in subject to know more. If so or in that case, you are just the reader I hoped to come across with my writing.
So dear reader, with the decision to learn or not to learn being yours to make, I will say this:

"Where the world is heading to is a digital and smart future."

Which means almost everything will be done by computers in fields such as teaching, marketing, music, medical, and many more. At that time, majority of the people that will thrive most are programmers, because they have the skill to guide computers to do tasks. Other careers might exist but will depend greatly on programmers.
I will also state that programming is neither easy nor hard. For me, it has been sweet; although sometimes hard, sometimes easy; but overall sweet. Nothing good is truly is easy, it's mainly about your passion, or will to do it.
This book is to touch on the basics of python programming languages as stated above.That is; basic things necessary to begin and build your programming skill.
Finally, the targeted readers of this book include Senior Secondary School Students and outright beginners in programming. These are people who have never in their lives programmed but can use and navigate on their computers. What is meant by "navigate", is to move from one directory or folder to another directory or folder; as well as type with the computer and open apps installed on the computer.

If you can do that, then you are ready to read chapter one, but if you do not know how to navigate your computer; then I strongly advise you learn the basics of operating a computer before proceeding.

CHAPTER 1:

Simple Brush up.

This chapter is going to brush up on things to know, but in a simplified manner.So, if you want to go in depth to it, I strongly advice you do your research. Let's begin with Programming.

Programming is literally you telling your computer things to do, in the computer language. You could tell your computer to do these things either through writing, speech or moving shapes. Now don't get confused by thinking, telling your digital personal assistant to do something is programming, because it's not. Why I say so is because you are not telling your computer to do the task. Rather, you're telling a software. Software? What now is a Software? don't worry, I will explain that later.

Back to Programming. For you to tell your computer to do things, you must speak the same language as the computer. For example, imagine a stranger walks up to you, and starts speaking Tashiro Language. Except for the few who can speak it, and the others would go the extra mile to pretend he is not talking to them (and later run away); others would be lost and smile at him. That's exactly how your computer reacts, or would react if it were nice.

Now you can tell your computer to do things, but only in a language that the computer can understand. So, what is the language that the computer can understand? The answer is Machine Language (because it is a language for machines. "Don't leave me (x3)").

Machine language (a.k.a. the computer's first language) which you may think is complicated a language with all the funny looking alphabets or numbers; although it really is complicated, it is not as complicated as you think. For example, relating to the funny looking Alphabets; Machine Language has none, Yes Machine Language has zero alphabets.

You're Probably thinking "Wait! So, it only has numbers?" Well you're right. Machine Language only has numbers. To be more specific, only has two numbers; which are 1 and 0. With that you can see that machine language isn't that complicated and might even be easier than our human languages (like English, or Japanese).

Now, I bet you are interested in machine language and would want to start learning it. What I can say to you is **Goodluck!**because although the language is not complicated, when using it, you'll be prone to errors; heavy errors, and also it is not easy to debug.

Lastly, the machine language is specific to one operating system. For instance, the machine language for windows is different from the machine language for

Macintosh. Therefore, if you want to tell your machine to do something in machine language, then you have to learn the machine language for that operating system so as to tell it what to do.
Also note that telling your Windows computer to do something in Machine Language is more time consuming than telling your MSDOS computer.
Meaning using a Graphical User Interface Operating System would be more time consuming than a Command line Interface Operating system if you are telling them things to do in machine language. I have a feeling some of you do not know what Graphical User Interface is nor Command line interface. Not to worry; I will also shine more light on them.
Next, you will be thinking; Is this what programmers do?How many machine languages is a novice programmer meant to know? and probably a few more questions I may not have thought of; but let me answer the few I had already listed.
First Question: is this what programmers do? the answer is yes; but not majority of modern programmers. What I mean is; unless you are a programmer that is directly involved in the stages of manufacturing an operating system or the computer itself,then there is no need to learn Machine Language. I know you have another question, but don't worry; I will answer the possible question in the next paragraph.
Second Question, how many machine languages is a noviceprogrammer meant to know? The answer is none. It is not compulsory for a programmer to know Machine Language. In the past, Programmers had to know machine Languages; so as to get their jobs done. At that time there were so many limitations, and it was very easy to make mistakes or error; Example was access to the computer itself, Programmers had to write or jot down their codes and ensure it works before telling/giving/typing it to the computer; you can imagine how frustrated they were. Along the line of history, Programmers decided they have to make their jobs easier; so they made a language that uses some English or human terms like Add, and many more; to communicate with the computer. The computer did not understand the raw language, so there was an interpreter to interpret the English or human-like languages to the computer language (machine language), This new language is what is known today as Assembly language. It made programming relatively easier, but still time consuming.
A few years later after Assembly language was made, more languages; that are more similar to human languages, were made; These languages were called High level languages, Example of High level Language are Java, C, C++, Python, and many more; funny thing is, more languages will still be made; some unnecessary, others unique.
With the brief insight on the history of programming language the next chapter will examine necessary jargons for a novice programmer.

CHAPTER 2

Jargons

The previous chapter dealt with the simplified history of programming (**if you want to go into details; like knowing the years it happened or the people involved in it; then please do your research**). This chapter will put in view the necessary jargons that a novice programmer needs to know. The itemized jargons are positioned with their individual explanations.

They include::

1. Text Editor: This is a software that is used to write codes. It has a feature to highlight the keywords of the programming language. Example of text editors are Sublime, Notepad++, and many others.
2. . Keywords: These are words the interpreter uses to recognize the structure of the program; There are 33 keywords in python, they are:
    1. and
    2. del
    3. or
    4. as
    5. for
    6. pass
    7. assert
    8. from
    9. raise
    10. break
    11. global
    12. return
    13. class
    14. if
    15. true
    16. continue
    17. not
    18. try
    19. def
    20. import
    21. while
    22. elif
    23. in
    24. with

25.else
26.is
27.yield
28.except
29.lambda
30.False
31. None
32.Finally
33. nonlocal

3. Interpreter/compiler: This is a software or a tool used to perform a task; The interpreter will transform your high-level code to Machine language, but it does it line by line, so the computer will understand, and execute it.
4. Run/ Execute: These are verbs used to represent perform. Meaning; rather than saying “Zack, perform the codes.” You can say “Zack, run your codes..”It is used to represent the performing of your code.
5. Code: This is the task for the computer, written down (on piece of paper) or typed (using a software tool, either a text editor or IDE).
6. IDE: This means Integrated Development Environment. This is a text editor (**a software tool**) that has a lot of features to enable an individual code, run, and debug a code; within it (**the software**).
7. Modules/Libraries: These are pre-written codes to perform a specific task, pre-written meaning already written; These pre-written codes were written in such a way to be very specific or custom. For instance, there is a pre-written module to make your computer speak. This same module has parts or methods for you to pick the voice, the rate of speed of the voice, or even the volume of the voice.
8. Methods: This is a collection of codes. A method has different parts, which are; The method signature and the body of the method.
9. Software: This is a collection of programs, images, and other related files that are all used to perform a task. Example: with Microsoft Word; Its icon or logo is a picture, while the app itself (**that is the winword.exe**) is a program.The app also has other documents necessary for its running or execution. Whenever the app runs, it uses all the related files, and documents, as well as the program; so as to enable the user create documents or and do other things with the document.
10. File Extension: These are suffixes (which usually come after a dot) that enable the operating system to know the nature of the file and how to treat it, or which app should or can open it. Example: on windows, .exe are executable files or apps, while .txt are text files.
11. Comment: These are notes that are not used by the interpreter.That is; the interpreter ignores this note.These notes are intended to be used by the

programmers to tell the programmer more about the code. There will be more on this in subsequent chapters

CHAPTER 3

Installing The tools.

Before going into typing, and crying (yes,you read right. I said crying, which you will understand later), We must learn to setup our tools.

The tools we need are:

1. Interpreter/Compiler
2. Text Editor/IDE
3. A Brain

For you to have come this far, then I believe you already have the 3rd tool installed; Now onto installing others.

Installing Python <Interpreter/Compiler>

There are different versions of python, but in this book, python 3.8.3 will be used because as at the time of writing this book, it is the latest version of python there is. The installation has been organized in steps.

Step1: Go to Google
Step2: Search "download Python 3.8.3"
Step3: The first link with Website address "www.python.org".
Step4: In the files section of the page, download the python version for your operating system and computer specification (implying; if your computer is a 32-bit, then download 32-bit; but if it is 64-bit, then download 64-bit), I strongly advise you download the executable installer (windows).
Step5: After downloading, open the installer.
Step6: Before you press install, tick/check the Add Python to path.
Step7: Press install

When you are done with the above, you might feel like nothing happened; So, to prove/check you have installed Python on your system. Then open your command prompt, and type:

python -–version

When the above is typed, you should be able to see the python version displayed; if you cannot; Then python is not installed and you should redo the above steps.

Installing Notepad++<Text Editor>

The text editor is selected as opposed to an IDE because I personally believe that is where beginners should start from, and its size is relatively smaller than its IDE counterpart (PyCharm).

Step1: Go to Google
Step2: Search for “Download Notepad++”.
Step3: Open the first link with the domain “notepad-plus-plus.org”.
Step4: choose the one at the top, under the Downloads section.
Step5: Download the installer for your computer specification (that is 32-bit for 32-bit, and 64-bit for 64-bit)
Step6: Open the installer.
Step7: Select the Installation Language.
Step8: Click Agree (meaning you agree to the terms and condition).
Step9: Click next, it will create a directory to install the app.
Step10: From the drop down beside “Select the type of install:”, I recommend you choose Custom.
Step11: Click on Next.
Step12: Check/tick “Create shortcut on desktop”.
Step13: Click Install.

Once the steps above are complete you will see an its icon (a green chameleon on a pencil, and a green paper-like diagram behind it) on your desktop.

Done with setting up the tool? The next chapter begins programming. I advise you switch on your pre-installed brain, and set it to **creative learning mode** because from here on, marks the beginning of your programming life.

CHAPTER 4

Data types.

Well, let’s say these are things or values; That you need the computer to use, either immediately or later. That’s one of the easiest ways I can put it, but I have a feeling I made you more confused; so I’ll use an illustration.
You have clothes, and your clothes range from Top, Bottom, and footwear. Examples of tops are Shirts, Vests, and a few others; Examples of Bottoms are shorts, trousers, amongst others; Example of footwears are socks, shoe, et cetera. When you buy new clothes or do laundry, do you just throw it on the bed?... Well, I do; but that is not where we are going.
You are not meant to throw it on the bed; you will rather arrange them into their respective closets as well as keeping underwear separate from tops.

Now to relate:

The Clothes represents data or values.

There are different categories of clothes, because each category has its own use. Example: Clothes have categories like Top, Bottom, Footwear.This is the same with data. There are different categories, because of different uses. These categories are Integers, Characters,Floats, etc..

These different categories of clothes, all have a closet where they are kept.In Programming, these closets are called Variables. Variables will be explained in the next chapter.

More Information on Data Types:

1. Integers: These are whole numbers, ranging from negative whole numbers to positive. Example of Integers: -1, -392930, 44444, 99999909, -484372.
2. Characters: These are single **symbols**, or is anything you can type in one keystroke, like a letter, a number, or a backlash.Example of Characters are: j, o, h, n, , 2, @.
3. Strings: This is a collection/sequence of Characters. That is, where 2 or more characters are gathered. It is a string.Example of Strings are: John, John2, John2@, John ate Potato.
4. Float: They are numbers with decimal points, because they are represented in a format called floating point. Example of Floats are: 1.22, 0.39, 9.094.
5. Lists: This refers to a sequence of values. Yes, just like string; except the values in a list could also be strings or other data types.In other words, a list is a list of items. Also, a list is enclosed by Square Brackets. Example of lists: [1, 10, 19], [‘John’,’love’, ‘Potato’].
6. Tuples: This is a sequence/collection of items, just like the List; but with a key difference;which is a tuple. A tuple is immutable (cannot be changed) and is surrounded by brackets. Example of Tuples are: (‘pot’, 1, 30, ‘Sponge’, Potato).
7. Boolean: Refers to a data type that has only two values which are True or False. Meaning, this data type is to know/check if a condition is true or if the condition is false.
8. Dictionary: A collection of values/items, which are called a key; These keys are mapped to another value/item, which are called values. Further explanation, In a real Dictionary (The large 300+ page book of words and its meaning), each word is mapped with its meaning.If you are looking for a word, you will get the meaning of the word.That is the exact concept with Dictionary Data Type. The word in the real dictionary is similar to the key in our Python Dictionary data Type, while the meaning of the words in the real Dictionary are similar to the value, in our Python Data Type. In Python Dictionary Data Type the association of a Key and Value, is called a key-value pair; or sometimes an item. Also, in the PythonDictionary Data Type,

the Dictionary is enclosed by Curly brackets, and the mapping is represented using colon.Example of Dictionaries:

A Dictionary of people and their age =
{‘paul’ : 26,’John’ : 39,’Susan’ : 25, ’Boma’ : 20}

A Dictionary of People and Their favourite food =
{‘Favour’: ‘Plantain’, ’Sharon’: ’Fish’, ‘Lorry’: ‘Banana’, ’Jack Sparrow’: ‘Potato’}.

That being said the next chapter will give insight on Variables. The Closet for Keeping the clothes(Data/values).

CHAPTER 5

VARIABLES

Having introduced this in the previous chapter, this chapter is a little dive into it. A variable is a container for storing values, data, or collection of data. You can store integers, characters, strings, lists, or any other data type into the variable. How to do this is by typing or writing (depends on where you are coding) in the below format:

name = value

The format above is how to declare a variable, and at the same time put a value or data into it. The left side, which has ‘name’, represents the name of the variable (there are rules for naming your variables, for which will be touched later on in the chapter). On the right side, which has ‘value’; represents the data you will put into the variable. This data could be any data type, that is integer, list, or any other data type.

Example of declaring variables:

One = 1
numbers = [1,2,3,4]
alphabet = [‘a’,’b’,’c’]

RULES TO NAMING VARIABLES

1. A variable name should not be a keyword.
2. It should not start with a number.
3. It should not have space.
4. It can contain underscore (_) {it is not compulsory to put underscore in the variable name, but if you want to, you can}
5. You can use uppercase, lowercase, or both.
6. It can not contain any other symbol except underscore.

Example of good variable’s names that meet the rules are:

1. NaMe
2. Did_he
3. aA_girl
4. potato
5. activated

Example of bad variable's names that do not meet the rules are:

1. did he?
2. A girl?
3. pot@to
4. for

When naming a variable, the rules stated must be met, and to help you code without always checking if your variable name meets the rules, you can make use of good practices. Yes,Programming has good practices, a few are listed below:

1. The name of variable should be related to what it is used for.Example, A variable used to store someone first name, will make complete sense if the variable name is **first_name**.
2. The name should not be unnecessarily long nor unnecessarily short.
3. The variable name should begin with a lowercase alphabet.

Well with the use of those rules and good practices, you will be able to program better without having problems with variables'name.

CHAPTER 6

OPERATORS AND OPERANDS

Operators are special symbols that represent mathematical calculations.example: addition, subtraction, etc..
The Operators symbols and actions are tabulated below:

SYMBOL	ACTION
+	ADDITION
-	SUBTRACTION
*	MULTIPLICATION
/	DIVISION
**	EXPONENTIAL
//	DIVISION[BUT ONLY TAKING INTEGER, WHILE THROWING AWAY REMAINDER OR DECIMAL]
%	MODULUS[THIS DISPLAYS THE REMAINDER ONLY]

Some Operators in use:

1. 1 //2 = 0
2. 3 ** 2 = 9
3. 5 % 2 = 1

In a situation for which you have multiple operators in a question, then you must know their order of priority. That is; which operator would run before the other. Example:

1. 2+1**5//2
2. 4-9//2**5

Unless you know which runs first or the order of the operators, you won't be able to solve the above.
In the Python Version we are using , which is Python 3.8.3, the order of operators are listed below:

Order based on Priority

1. Parenthesis
2. Exponential
3. Division
4. Multiplication
5. Addition
6. Subtraction

The most popular acronym amongst programmers to recall the priority is **PEDMAS**. With the priority established, we can solve question 1 and 2.

SOLVING QUESTION 1

1. 2+1**5//2 **The first to run will be Exponential, which is 1**5; and we will get 1.**

   =2+1//2 **The next to run, will be Division; which is 1//2, which is 0**
   =2+0 **And finally, the last operator which is addition.**
   =2 **The answer is then 2.**
   You can confirm the solution on your python console, to do that you should check out the repetitive action page.
2. 4-9//2**5 **The first to run will be exponential, which is 2**5; which is 32.**

   = 4-9//32 **The next to run will be division, which is 9//32; which is 0.**

   = 4 – 0 **And finally, the last to run will be the subtraction operator.**

   =4 **The answer is then 2.**

You can confirm the solution on your python console, to do that you should check out the repetitive action page.

Now we are done with Operators, and Operands. In the next chapter, we will handle conditional statements.You don't need to go read on it yet; just sleep and practice what you have read until now.

.

CHAPTER 7:

CONDITIONAL/LOGIC STATEMENT

In order to write useful programs majority of time, we will need to check conditions and perform an action based on the condition. The condition statement gives us the ability to do so.
The conditional statement is a statement that checks if a condition is met or true, and if it is; would run a block of codes. But if not, the next statement would be run.
Let me give a simple example:
John said Paul should buy bolee, and fish; and should also buy middle fish but if no middle fish he should buy only bolee.
To represent this in Pseudocode:
>>Buy Bolee
>>if (fish_head_available == 'YES'):
>>buy fish_head
>>end

In the pseudocode above, the conditional statement is the if statement.

What about a situation where you are to also perform an action if the condition is not meant? Do not worry, that is where ‘else’ comes in.
Eze said Paul should buy bolee, and fish; and should also buy middle fish if not head fish.
To represent this in Pseudocode:
>>Buy Bolee
>>if (fish_head_available == ‘YES’):
>>buy fish_head
>>else:
>>buy middle_fish
>>end

Now, based on the examples, the most simple form of the conditional statement is the ‘if’ statement. Syntax below:
>>if (condition):
    >>blocks_of_code

In the above, the parenthesis immediately beside the ‘if’ is the condition that must be met for the blocks of code to run, but if condition is not met, the blocks of code would not run.

An example of the above:
If (x > 0):
Print(‘x is positive’)

If (gender == ‘male’)
Print(‘you male’)

The syntax for a situation you want to run a block of code when the condition is met and when it is not met. The syntax below:
>>if (condition):
>>blocks_of_code_1
>>else:
>>blocks_of_code_2
In the above, it is the same syntax with an added else statement. If the condition is met, the blocks_of_code_1 is run, while the else and blocks_of_code_2 is skipped. If the condition is not met, then the blocks_of_code_1 is skipped; while the blocks_of_code_2 is run.
Example of the above syntax:
If (x > 0):
Print(‘x is positive’)

```
    Else:
    Print('x is negative')


    If ( gender == 'male')
    Print('you male')
Else:
Print('You female')
```

How about a situation, where you have two or more conditions to check? There are actually various ways to do this, but I will reveal the way I do it. I make use of the 'elif' statement comes in.
Elif statement is more like an 'else if', with that you have a vague understanding. The syntax below:

```
    >>if (condition):
    >>blocks_of_code_1
    >>elif (condition2):
    >>blocks_of_code_2
    >>else:
    >>blocks_of_code_3
```

From the above, you can tell the elif can contain the other condition you want to check. You can add as much 'elif' statement you need. Examples below:

```
    If (x > 0):
    Print('x is positive')
    Elif (x == 0):
    Print('x is zero')
    Else:
    Print('x is negative')


    If  (gender == 'male'):
    Print('you male')
    Elif (gender == 'female'):
    Print('you female')
    Elif (gender == 'not say'):
    Print('you do not want to say.')
    Elif (gender == 'other'):
    Print('there is more?')
Else:
Print('choose')
```

From the examples you can tell that you can add multiple amounts of 'elif' statements.

Practice Question:
Write a block of codes to check if a number is an odd number or even number or simply just zero.

CHAPTER TEST QUESTIONS (**THERE IS NO "ONE CORRECT WAY", SO TRY YOUR BEST**):

1. Write a code that checks if the user input is 1, or 2, or 3, or 4, or 5, or greater than 5, or less than 1.
2. Write a code that checks if the age, the user inputted is that of an adult or minor.

FINAL
We are done with this chapter, the next chapter is on loops, that is a way to repeat an action or iterate through a list or variable.

CHAPTER 8:
LOOPS

For this chapter, let us start with the problem.
Say you want to check through your list for school to confirm you added shoes to it. With your current skillset so far, you would not be able to do it.So, let us add the skill that would. That skill is called loops.
In Python, there are only two loops:-

1. While loops
2. For loops

1. While Loops: Are loops that repeat when a condition is met. A real life case could be: You want chocolate Ice cream, your dad goes in to the store to buy. You told him you will not take anything but chocolate; he agrees and goes back in. Your dad comes back with vanilla ice cream; you refuse to take it,

and tell him to go back and try again; he goes back, and comes out with strawberry, you still refuse; and tell him to go back and try again.

From the above, your dad would repeatedly go in and out, until he actually buys the chocolate Ice cream. To represent it in code:

```
>>while ice_cream != 'chocolate':
>>print('Go back, and try again.')
>>print('Thank you.')
```

The first line is for the condition; the dented lines are the actions that would be repeated until the condition is met. Once it is met the third line runs.

I know, this isn't how Nigerians fathers are; but let us just use it or this example.

2. For Loops: These loops are used to iterate a sequence.Example: lists, dictionaries, etc. A real life example is when reading out your school list to your parent; you must read each item out loud.

   The code representation:

```
>>for item in school_list:
>>print(item)
>>print('that is all')
```

   The first line selects an item from the list, the second line displays the item; the process is repeated until all the items have been chosen. Then the third line runs.

BREAK AND CONTINUE KEYWORDS

1. The Break Keyword is used to exit or end a loop.
2. The continue keyword is used to quit the current/ immediate block of code, then return to the parent loop.

PICTURE EXPLAINING PAREN LOOP COMES HERE.

Code samples below:

```
>>count = 0
>>While True:
>>print(count)
>>count += 1
```

```
>> if count >= 3:
>>break
>>
>>for number in range(10):
>>print('We are here now.')
>>if number % 2 == 0
>>print(number)
>>continue
>>print('this line skipped')
```

Infinite loop: These are loops without an ending or a breakpoint, it just loops till the execution is closed or interrupted.

PRACT TEST

1. Write a code to display all your items on your school list except for the 7th item.
2. Write an infinite loop that displays 'sup.'

CHAPTER 9:

FUNCTIONS

This is a block of codes that perform a task. The use of functions enables people reuse code rather than have to type it over and over again.

A better way to understand this is an example below:

We have all done the dishes before [that is,washed plate], I don't know about you, but for me it isn't something I would want to repeat everyday of my life.

So, What do I do? I teach my younger one how to wash plates and call her anytime I have to.

THOSE PICTURES OF MANIQUIEN AND TEXT SAYING 'BREINS'

Well, that's what I would've done if she were still a child.

The above paragraphs should give you a general understanding of functions; now let's trek a little deeper, My transport fare finished when writing this chapter. **GOOGLE HOW TO INSERT EMOJIS**

PARTS OF A FUNCTION
Below is a code example of how to write a function in python.

```
>>def function_name(parameter1,parameter2):
>>print('our first function')
```

A function has the following parts.

1. Head
2. Body
3. Parameters

1. Head: This is the line that starts with 'def' and ends with a column.This line contains the function's name and parameters. Naming a function is similar to naming a variable; therefore the same good practices for naming a variable can be used in naming a function.Using our function example earlier, the head is line one, which reads:

   ```
   >>def function_name(parameter1,parameter2):
   ```

2. Body: this is the part of the function that actually performs the task. Meaning, the line or block of code that performs a task. Using our example of functions, the body is a line of code, which is:

   ```
   >>print('our first function')
   ```

3. Parameter: Also called arguments, are data given to the function to help customize the tasks. Example let's say you want the computer to greet the user by his or her name. You would need the user to input their name, then make the computer to print it out.

   The code version below:

```
>>def greet_user(username):
>>print('Good day', user_name)
>>name = input('Input your name')
>>greet_user(name)
```

The above code would print out the below for me:
Good day Lawrence
The above code would print out the below for Loxxy:

Good day Loxxy

For a situation you want your function to return data after performing actions on it; you make use of the return statement. An example of the return statement in use: Let's say, you want to get someone's age 2 years from now, by adding 2 to their current age, and then display it on the screen.
Code below:

```
>>def get_new_age(age):
>>new_age = age + 2
>>return new_age
>>
>>age = input('Input your age')
>>new_age_ = get_new_age(age)
>>print('Your age 2 years from now is ', new_age_)
```

From the example above the third line is the line that returns the data and exits or ends the function. Whenever a return statement is encountered, the computer returns the data asked for and returns to the previous line before it entered the current block of code. In the example, it left the function and returned to the main block of code, that is line 6.
Note: If you are using the return statement to return data, then you must use a variable to store the returned data. Line 6 does this.
Why do you need to do this? Because the place where the function was called would be replaced by the returned data. From the above example:
Line 6: >>new_age_ = get_new_age(age)
Line 6 after data returned(using 3 as current age): >>new_age_ = 5

If this data isn't stored, it'll only be there; unable to be used.
A simple way to understand:

You are walking down a path labelled one to ten.

At point 1 you decide to build a tool making machine.You continue down the path and finished building it at point 3.

At point 5, you need a hammer, so you call the tool making machine and state the specific tool you want as a parameter orargument; in this case it is a hammer. You then wait patiently for the tool to be made and delivered.

Once the machine is done it returns the hammer to point 5 because that was the place it was called. The machine drops the tool on the floor. You cannot use the tool until you pick it up or put it in your tool bag.

Leaving the tool on the floor is the same as not storing the returned data in a variable. While, picking up the hammer or dropping it in the tool bag is the same as storing returned data into a variable.

You have made it to the end of this chapter, congratulations; It ain't easy but you did it. The next chapter is the final, and that chapter talks on what errors are and various errors; what are modules, what is Pip, and how to use it. Before moving to the next chapter practice everything you have learned so far.
GOODLUCK.

CHAPTER 10:
ERRORS, LIBRARIES, AND PIP

What are Errors?
These are problems encountered by the computer while running your code. This problem could stop the execution or the running of your code. Whenever this happens python tries it's best to tell you why the error occurred, and suggests a way to fix it.
In programming there are 2 major types of errors:

1. Syntax Errors: These are errors gotten when the rules of the language, are not obeyed. For instance, not putting semicolon after the function head.

   In code:
   >>def function_name(parameter)
   >>print('It would flash an error')

The above would flash an error because of the absence of a semicolon.

2. Logical or Semantic errors: These are errors that meet the rules of syntax, but do not do what the programmer intended or desired. Example is performing a function on the wrong variable.

    Let us say:

    i. You want to get the user's current age.

    ii. Use the current age to get the age 2 years from now, by adding 2.

    iii. Use the current age to get the age 5 years from now, by adding 5.

The semantic error could arise, when you use the age 2 years from now, to get the age 5 years from now, by adding 5 to it.

There are a few other types of errors, like FileNotFound
Error, ModuleNotFoundError, and many more. You would encounter them in your life as a programmer.
Python tries to point out errors, but finds it harder to point out semantic errors because the rules of syntax are correct. You as a programmer would have to find it out yourself.
For every error except semantic, you can make use of the try and except to handle it.
What I mean is, majority of errors when encountered, would crash the app [close or stop the app]; So, to avoid your app from crashing, you use a try and except.

Think of it like this:
Try to do this task,
    Except something bad happens,
    Then do this task
Let's do the above in code:

```
>>try:
>>print('Task A would be done')
>>except error:
>>print('Task B would be done.')
```

The above would help stop the app from crashing, if it experiences an error.

LIBRARIES:
These are set of useful functions that eliminate the need for writing codes from scratch.

Let's say I have to do the dishes; my baby sister isn't around and I heard of a company producing free robotic maids. No joke that's a dream come true. Back to the example - I would use my phone to search for the robotic maid and the store, when I find it; I would place an order to be delivered to my home.
Once I receive it, I use my phone to unpack and activate it. When I'm done, I would have a robotic maid that does the dishes for me. I would not need to always go to the sink turn on the tap nor use the soapy sponge to rub against the dirty plates, I'd just call the robotic maid and she does it for me.

From the above case:
The Robotic Maid is equivalent to a library.
The Phone is equivalent to Pip.
I am equivalent to you.
The Store is equivalent to PYPY

PIP:
This is python's package manager. It searches for libraries, installs the libraries, and uninstalls libraries. Similar to the phone that searches for the robotic maid and place orders.
How to use pip to search for a library: For this we are using pyttsx3.This is a library used to make your computer speak.

There are different ways to use pip I will only shed light on:

i. How to use pip to search.
ii. How to use pip to install.

HOW TO USE PIP TO SEARCH

1. Open Command Prompt.
2. Type the library you want to search for in the below format:

```
>>pip search <library_name>
```

Example: pip search pyttsx3
The above example would search and display a list of libraries with similar names to the library searched. But if it didn't find anything it wouldn't display.

YOU NEED INTERNET CONNECTION TO SEARCH OR INSTALL LIBRARIES FROM THE INTERNET USING PIP.

HOW TO USE PIP TO INSTALL A LIBRARY:

1. You search for the library you want to install.
2. Note down or copy the library's name.
3. Write the library you want to install using the below format.

>>pip install <library_name>
Example: pip install pyttsx3

Some lines would display and it would begin downloading. Then, once done with downloading it would begin installing.
Once the process is over, you can confirm if the process worked by using the 'import <library>' statement at the python terminal or command prompt.
If there is no error then it worked.

If you have reached this line, then you have made it to the end of the book; and your programming journey has literally begun. The question now is: **"what will you use this skill for?"** Would you hang it somewhere? Or build life impacting softwars and simultaneously grow your skill.
Whatever choice you make, I want you to know it is 100% yours..
Do stay safe and God bless.