Chapter - 3

INTRODUCTION TO PYTHON

Introduction to Python

- ❖ Python is a widely used general-purpose, high level programming language.
- ❖ It was initially designed by **Guido van Rossum in 1991** and developed by Python Software Foundation.
- ❖ It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
- ❖ Python is a programming language that lets you work quickly and integrate systems more efficiently.

Features of Python

Python is object-oriented	Structure supports such concepts as polymorphism, operation overloading and multiple inheritance.
Indentation	Indentation is one of the greatest feature in python
It's free (open source)	Downloading python and installing python is free and easy

Features of Python

It's Powerful	• Dynamic typing • Built-in types and tools • Library utilities • Third party utilities (e.g. Numeric, NumPy, sciPy) • Automatic memory management
It's Portable	• Python runs virtually every major platform used today • As long as you have a compatible python interpreter installed, python programs will run in exactly the same manner, irrespective of platform.
It's easy to use and learn	• No intermediate compile • Python Programs are compiled automatically to an intermediate form called byte code, which the interpreter then reads. • This gives python the development speed of an interpreter without the performance loss inherent in purely interpreted languages. • Structure and syntax are pretty intuitive and easy to grasp.

Features of Python

Interpreted Language	Python is processed at runtime by python Interpreter
Interactive Programming Language	Users can interact with the python interpreter directly for writing the programs
Straight forward syntax	The formation of python syntax is simple and straight forward which also makes it popular.

Installation

- There are many interpreters available freely to run Python scripts like IDLE (Integrated Development Environment) which is installed when you install the python software from http://python.org/downloads/
- Steps to be followed and remembered:

Step 1: Select Version of Python to Install.

Step 2: Download Python Executable Installer.

Step 3: Run Executable Installer.

Step 4: Verify Python Was Installed On Windows.

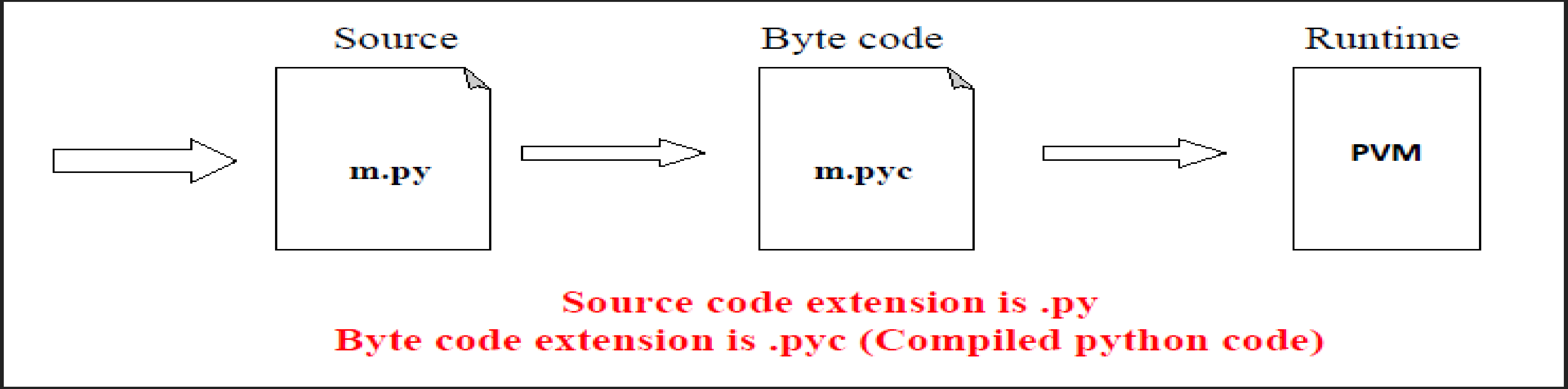
Step 5: Verify Pip Was Installed.

Step 6: Add Python Path to Environment Variables (Optional)

Working with Python

Python Code Execution:

- **Python's traditional runtime execution model:** Source code you type is translated to byte code, which is then run by the Python Virtual Machine (PVM).
- Your code is automatically compiled, but then it is interpreted.


Source
m.py
Byte code
m.pyc
Runtime
PVM
Source code extension is .py
Byte code extension is .pyc (Compiled python code)

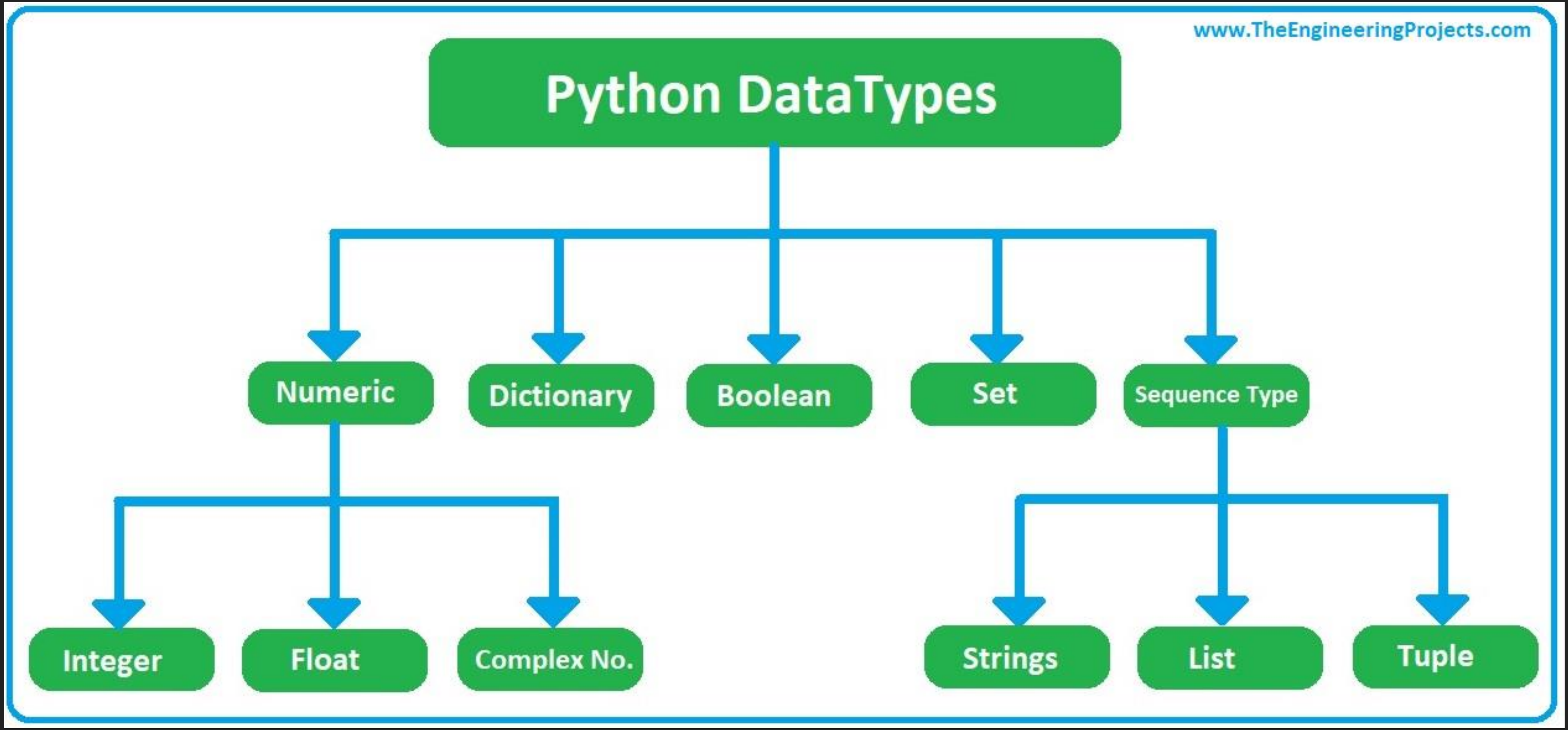
Working with Python

There are two modes for using the Python interpreter:

- Interactive Mode
- Script Mode

Variables

- There are a certain rules that we have to keep in mind while declaring a variable:
- The variable name cannot start with a number. It can only start with a character or an underscore.
- Variables in python are case sensitive.
- They can only contain alpha-numeric characters and underscores.
- No special characters are allowed. Python allows you to assign a single value to several variables simultaneously.
- For example − a = b = c =1
- Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.
- You can also assign multiple objects to multiple variables.
- For example − a,b,c = 1,2,"john".


www.TheEngineeringProjects.com
Python DataTypes
Numeric
Dictionary
Boolean
Set
Sequence Type
Integer
Float
Complex No.
Strings
List
Tuple

Basic Data Types

Integer

12

Integers are used to represent whole number values.

Ex, x = 10

Float

1.23

Float data type is used to represent decimal point values.

Ex, x = 2.5

Complex

$a+ib$

Complex numbers are used to represent imaginary values.

Ex, x = 2 + 3j

Boolean

Boolean is used for categorical o/p, since the o/p of Boolean is either true or false.

Ex. X = 1> 2

Why integer when there is float ?????

Python Operators

- ❖ Arithmetic operators
- ❖ Assignment operators
- ❖ Comparison operators
- ❖ Logical operators
- ❖ Identity operators
- ❖ Membership operators
- ❖ Bitwise operators

Python Arithmetic Operators

X = 10 & Y = 6

Operator	Name	Example	Output
+	Addition	x + y	16
-	Subtraction	x - y	4
*	Multiplication	x * y	60
/	Division	x / y	1.66
%	Modulus	x % y	4
**	Exponentiatio n	x ** y	1000000
//	Floor division	x // y	1

Python Assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x = +3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3

Python Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Operator	Name	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Operator	Name	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Operator	Name	Example
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Python Input

- ❖ input()
- ❖ input().split(separator, maxsplit)

Python Branching

- ❖ If statement
- ❖ Elif Statement
- ❖ Else statement
- ❖ Short hand if
- ❖ Short hand if – else

If statement

- The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block.
- The condition of if statement can be any valid logical expression which can be either evaluated to true or false.
- The syntax of the if-statement is given below.

```
if expression:
          statement
```

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even")
```

Output

The if-else statement

- The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition.
- If the condition is true, then the if-block is executed. Otherwise, the else-block is executed.

```
if condition:
    #block of statements
else:
    #another block of statements (else-block)
```

```
num = int(input("enter the number?"))
if num%2 == 0:
    print("Number is even...")
else:
    print("Number is odd...")
```

Output

The elif statement

- The elif statement enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them.
- We can have any number of elif statements in our program depending upon our need. However, using elif is optional.
- The elif statement works like an if-else-if ladder statement in C. It must be succeeded by an if statement.

```
if expression 1:
   # block of statements

elif expression 2:
   # block of statements

elif expression 3:
   # block of statements

else:
   # block of statements
```

```
number = int(input("Enter the number?"))
if number==10:
   print("number is equals to 10")
elif number==50:
   print("number is equal to 50");
elif number==100:
   print("number is equal to 100");
else:
   print("number is not equal to 10, 50 or 100");
```

Output

Short Hand If

- If you have only one statement to execute, you can put it on the same line as the if statement.

```
if a > b: print("a is greater than b")
```

Short Hand If ... Else

- If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

This technique is knowns **Ternary Operators**, or **Conditional Expressions**.

```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

The pass Statement

- if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

```
a = 33
b = 200

if b > a:
  pass
```

Python Iteration

- ❖ **While loop**
- ❖ **For loop**
- ❖ **Continue and Break**

While Loop

- The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop.
- It can be viewed as a repeating if statement. When we don't know the number of iterations then the while loop is most effective to use.

```
while expression:
    statements
```

For Loop

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

For Loop – range()

- To loop through a set of code a specified number of times, we can use the range() function,
- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Break Statement

- **1. Continue Statement -** When the continue statement is encountered, the control transfer to the beginning of the loop. Let's understand the following example.
- **2. Break Statement -** When the break statement is encountered, it brings control out of the loop.

Python Game Project

1. **Dice Rolling Simulator**
2. **Guess the Number**
3. **Mad Libs Generator**
4. **Text-Based Adventure Game -A complete text game, the program will let users move through rooms based on user input and get descriptions of each room. To create this, you'll need to establish the directions in which the user can move, a way to track how far the user has moved (and therefore which room he/she is in), and to print out a description. You'll also need to set limits for how far the user can move. In other words, create "walls" around the rooms that tell the user, "You can't move further in this direction."**
5. **Hangman**

References

- **https://www.w3schools.com/python/default.asp**
- https://www.javatpoint.com/python-tutorial
- https://www.tutorialspoint.com/python/index.htm
- Wesley J. Chun. “Core Python Programming - Second Edition”, Prentice Hall
- John V Guttag. “Introduction to Computation and Programming Using Python”, Prentice Hall of India