Python Notes - Week 3

Date:- 28/02/2022

Introduction To Python

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports all the OOPS concepts, multi-threading, exception-handling, and automatic memory management which help in solving real-world problems and building applications to solve these problems.

**History of Python**:- It was invented by Guido van Rossum in 1991 at CWI in the Netherlands. The idea of Python programming language has taken from the ABC programming language and it's name has been derived from the popular BBC comedy show "Monty Python's Flying Circus".

**Features of Python:-**

a) Simplicity

b) Free and Open Source

c) Portability

d) Extensive support of third-party libraries

e) Object-Oriented Language

f) Dynamic Memory Allocation

**Python Interpreter-** An interpreter is a computer program that we need to run our python code or scripts. It directly executes the instructions written in python programming language, without requiring them previously to have been compiled into a machine language program. So unlike Java, Python uses an interpreter.

**Python Installation-** Most versions of Ubuntu come with Python pre-installed. To check the version of Python we can type

-> python3 -V

a) To make sure that our versions are up-to-date, we update our local package index using "sudo apt update"

b) Then upgrade the packages installed on your system to ensure you have the latest versions using "sudo apt -y upgrade"

c) Once the process is complete, we can check the version of Python 3 that is installed in the system by typing "python3 -V"

**Python IDE(PyCharm) Installation-** The IDE typically provides code editor, compiler/ interpreter and debugger in one GUI (Graphical User Interface). It encapsulates the entire process of code creation, compilation and testing. PyCharm has been developed by Jetbrains and is specifically developed for Python. It provides two versions for download, the Community version which is free and the paid Professional version.

Steps for Installing PyCharm Community Version using Command Line

a) In our terminal window, we type the following command for installing PyCharm-> sudo snap install pycharm-community --classic

b) Once installed we can start PyCharm from the terminal using the command-> pycharm-community or by selecting the Pycharm icon directly from the menu.

Besides PyCharm, we can also make use of "Jupyter Notebook" to write our Python scripts which is an open-source web application that allows you to create and share documents that contain live code. Jupyter Notebook is built off of Python and the Kernel runs the computations and communicates with the Jupyter Notebook front-end interface. It is used for data cleaning and transformation, data visualization, machine learning, and much more.

**Applications of Python**

1) Data Science

2) Data Mining

3) Desktop Applications

4) Console-based Applications

5) Mobile Applications

6) Software Development

7) Artificial Intelligence

8) Web Applications

9) Enterprise Applications

10) 3D CAD Applications

11) Machine Learning

12) Computer Vision or Image Processing Applications

**Some Common Functions-**

a) **Python print() Function-** The print() function displays the given object to the standard output device (screen). Syntax of print function-

-> print(\*objects, sep=' ', end='\n')

b) **Input Function**- It is used to take input from the user.

Example-

name = input("Enter a name of student:")   
print("The student name is: ", name)

Basics of Python

**Keywords in Python-** Python Keywords are special reserved words that convey a special meaning to the interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as a variable name, function name or anything else. In total there are 3 keywords in Python:-

True False None and as

asset def class continue break

else finally elif del except

global for if from import

raise try or return pass

nonlocal in not is lambda

1) True - It represents the Boolean true, if the given condition is true, then it returns "True". Non-zero values are treated as true.

2) False - It represents the Boolean false; if the given condition is false, then it returns "False". Zero value is treated as false

3) None - It denotes the null value or void. An empty list or Zero can't be treated as None.

4) and - It is a logical operator. It is used to check the multiple conditions. It returns true if both conditions are true.

5) or - It is a logical operator in Python. It returns true if one of the conditions is true.

6) not - It is a logical operator and inverts the truth value.

7) def - This keyword is used to declare the function in Python. If followed by the function name. Example-

| def my\_func(a,b):   c = a+b   print(c)  my\_func(5,9) |
| --- |

8) class - It is used to represent a class in Python. The class is the blueprint of the objects. It is the collection of the variables and methods. Consider the following class.

| class Democlass:   #Variables  def function\_name(self):   #statements |
| --- |

9) continue - It is used to stop the execution of the current iteration. Consider the following example.

| a = 0  while a < 4:   a += 1   if a == 2:   continue   print(a) |
| --- |

10) break - It is used to terminate the loop execution and control transfer to the end of the loop. Consider the following example.

| for i in range(5):   if(i==3):   break   print(i)  print("End of execution") |
| --- |

11) If - It is used to represent the conditional statement. The execution of a particular block is decided by if statement. Consider the following example.

| i = 18  if (1 < 12):  print("I am less than 18") |
| --- |

12) elif - This Keyword is used to check the multiple conditions. It is short for else-if. If the previous condition is false, then check until the true condition is found. Condition the following example.

| marks = int(input("Enter the marks:"))  if(marks>=90):   print("Excellent")  elif(marks<90 and marks>=75):   print("Very Good")  elif(marks<75 and marks>=60):   print("Good")  else:   print("Average") |
| --- |

**Python Variables-** A variable is a named location used to store data in the memory. It can be thought of as a container that holds data that can be changed later in the program. For example,

| number = 10 |
| --- |

In Python, we don't actually assign values to the variables. Instead, Python gives the reference of the object(value) to the variable.

We can change the value of the variable as per our requirement and when needed. For eg

| number == 21 |
| --- |

If we want to assign the same value to multiple variables at once, we can do this as:

| var\_1 = var\_2 = var\_3 = "Same Text Given" print (var\_1) print (var\_2) print (var\_3) |
| --- |

If we want to assign values to multiple variables in a single line, we can do this as:

| a, b, c = 5, 3.2, "Hello" print (a) print (b) print (c) |
| --- |

-> **Variable Types**

1) Local Variable- Local variables are the variables that are declared inside the function and have scope within the function. Consider the example

def addNos():   
 # Defining local variables.   
 var\_1 = 12   
 var\_2 = 52   
 result = var\_1 + var\_2   
 print("The sum is:", result)   
   
addNos() # Calling a function

2) Global variables- They can be used throughout the program, and its scope is in the entire program. We can use global variables inside or outside the function. A variable declared outside the function is the global variable by default. Python provides the global keyword to use global variables inside the function. If we don't use the global keyword, the function treats it as a local variable. Example-

| glob\_var = 122    def globFunction():   global glob\_var # printing a global variable   print(glob\_var)     glob\_var = 'Number has been changed to string' # modifying a global variable   print(glob\_var)    globFunction()  print(glob\_var) |
| --- |

**Python Constants-** A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later. In Python, constants are usually declared and assigned in a module. Here, the module is a regular python file with .py extension containing variables, functions, etc which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words. Consider an example-

Create a constants.py:

| PI = 3.147 DEMO\_CONST = 29.5 |
| --- |

Create a main.py:

| import constants # import keyword is being used to import or use the entities defined in constants.py file  print(constants.PI) print(constants.DEMO\_CONST) |
| --- |

**Python Literals-** Literal is raw data given to a variable or constant. It can be-

a) Numeric Literals- It belongs to 3 different numerical types: Integer, Float, and Complex. For eg

| var\_1 = 19 # Integer Literal  var\_2 = 15.23 # Float Literal  var\_3 = 7 + 8.54j # Complex Literal- A complex value has two parts real part and imaginary part(denoted by j) |
| --- |

b) String literals- String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes to create a string. Eg- "Robert", "Hello..How are you?"

-> Single-line String- Strings that are terminated within a single-line are known as Single line Strings.

Example:

str\_text\_1 = 'hello'

-> Multi-line String - A piece of text that is written in multiple lines is known as multiple lines string. There are two ways to create multiline strings:

1) Add a black slash at the end of each line.

Example:

| str\_text\_1 = 'hello\  george'  print(text1) |
| --- |

2) Using triple quotation marks:-

Example:

| str\_text\_2 = '''''welcome  to the organization! ''' |
| --- |

c) Boolean literals - A Boolean literal can have any of the two values: True or False. Example-

| x = (1 == True) y = (1 == False) a = True + 6 b = False + 21  print("x is", x) print("y is", y) print("a:", a) print("b:", b) |
| --- |

d) We can also specify a special literal known as "None" which represents Null values in Python.

**Python Comments-** Comments are very important while writing a program. They describe what is going on inside a program, so that a person looking at the source code does not have a hard time figuring it out. In Python, we use the hash (#) symbol to start writing a comment.

Example-

# Printing out a Float Value  
float\_var = 11.98  
print(float\_var)

The above type of comment is also known as single-line comment.

To have comments that extend up to multiple lines, we can use hash(#) symbol at the beginning of each line like

| # Hello # Good Morning # What's the agenda for today? |
| --- |

-> Another way of doing this is to use triple quotes, either ''' or """. These triple quotes are generally used for multi-line strings. But they can be used as a multi-line comment as well.

**Python Indentation-** Indentation in Python refers to the whitespaces that the developer must provide while writing a block of code as use of curly braces is not there in Python unlike other languages. Generally, four whitespaces are used for indentation and must be consistent throughout that block. Eg-

| if var\_1 = "New Text":  print(True) |
| --- |

**Practice Questions**

Q.1 Python program which accepts the user's first and last name and prints them in reverse order with a space between them.

**Sol:-**

| first\_name = input("Enter your first name: ") last\_name = input("Enter your last name: ")  print("Hello", last\_name, first\_name) |
| --- |

**Output:-**

| Enter your first name: George Enter your last name: Adam Hello Adam George |
| --- |

Q.2 Python program which accepts the radius of a circle from the user and compute the area

**Sol:-**

| PI = 3.147 # Constant Variable  # Converting the string value into integer value using built-in int function circle\_radius = int(input("Enter the radius of your circle: ")) area = (PI \* circle\_radius \*\* 2) print("Area of circle is ", area) |
| --- |

**Output:-**

| Enter the radius of your circle: 4 Area of circle is 50.352 |
| --- |

Q.3 Python program to swap two variables without using Third Variable

**Sol:-**

| num\_val\_1 = int(input("Enter the first number: ")) num\_val\_2 = int(input("Enter the second number: "))  print("\nBefore Swapping") print("First Number: ", num\_val\_1, "\nSecond Number: ", num\_val\_2)  num\_val\_1 = num\_val\_1 + num\_val\_2 num\_val\_2 = num\_val\_1 - num\_val\_2 num\_val\_1 = num\_val\_1 - num\_val\_2  print("\nAfter Swapping") print("First Number: ", num\_val\_1, "\nSecond Number: ", num\_val\_2) |
| --- |

**Output:-**

| Enter the first number: 13 Enter the second number: 3  Before Swapping First Number: 13  Second Number: 3  After Swapping First Number: 3  Second Number: 13 |
| --- |

Q.4 Python program to calculate simple interest

**Sol:-**

| principle\_amt = int(input("Enter the principal amount: ")) interest = int(input("Enter the rate of interest: ")) time = int(input("Enter the time period: "))  simple\_int = (principle\_amt \* interest \* time) / 100 print("Simple Interest is ", simple\_int) |
| --- |

**Output:-**

| Enter the principal amount: 2000 Enter the rate of interest: 6 Enter the time period: 2 Simple Interest is 240.0 |
| --- |

Q.5) Python program to find the maximum of two numbers

**Sol:-**

| var\_1 = int(input("Enter first number: ")) var\_2 = int(input("Enter second number: "))  if var\_1 > var\_2:  print("Maximum number is ", var\_1) else:  print("Maximum number is ", var\_2) |
| --- |

**Output:-**

| Enter first number: 34 Enter second number: 54 Maximum number is 54 |
| --- |

Q.6 Python program to check whether the given number is Armstrong number or not

**Sol:-**

| num\_1 = int(input("Enter a number: ")) temp = num\_1 digits\_sum = 0  while temp != 0:  rem = temp % 10  digits\_sum += rem \*\* 3 # \*\* denotes exponent operator as it calculates the first operand power to the second operand.  temp = temp // 10 # (//) represents floor division. On using this for division of two nos, it returns a whole number not a decimal value  if num\_1 == digits\_sum:  print(num\_1, "is an Armstrong Number") else:  print(num\_1, "is not an Armstrong Number") |
| --- |

**Output:-**

| Enter a number: 153 153 is an Armstrong Number |
| --- |