**PYTHON**

**Variables –**

* Variable name consist of latters, numbers and underscores. It can’t start with numbers.
* Python is case sensitive language, so ‘A’ and ‘a’ are two different variables.
* No need to define the variable before using them like C++.
* In python, variables don’t have specific type. So, you can assign string to variable and later integer to the same variable.
* Variables can be reassigned as many times as you want.

**Input and Output Functions –**

* To accept input from user ‘input’ function is used
* ‘print’ function is used to produce output.

**Data Structures –**

* Lists are created using square brackets with commas separating them.
  + A = [1,85,36,7,94]
* Tuples are similar to lists except they are created using parentheses instead of brackets and then are immutable (can’t be changed) unlike lists.
  + Words = (‘Hi’, ‘Hello’, ‘Bye’)
* Dictionaries are data structures used to map arbitrary keys to values. Each element in dictionary is represented by ‘key:value’ pair. They are created using curly brackets and are mutable.
  + Ages = {“Dave”:24, “Mary”:42, “John”:58}
* Sets are data structures similar to lists but created using curly brackets. Also sets do not support indexing like list and they can’t contain duplicate elements. The way they are stored, its faster to check whether an item is part of set or not using ‘in’ function.
  + Set = {1,6,7,9,4}
* Which data structure should be used 🡪
  + Dictionary – when you need a logical association between a key:value pair.
  + List – when you need simple, iterable collection of data that is modified frequently.
  + Set – when you need uniqueness for the elements.
  + Tuple – when your data can’t be changed.

**Control Statements –**

**If statement –**

You can use if statement to run code if certain condition holds. Python uses indentation (white spaces at the beginning of a line) to delimit blocks of code.

**Else statement –**

An else statement follows if statement and contains code that is called when the if statement evaluates to False.

**Elif statement –**

The elif statement is shortcut to use when chaining if and else statements. A series of if or elif statement can have a final else block, which is called if none of the if or elif expression is true.

**While loop –**

Code inside while loop is repeatedly executed as long as the condition holds true. To end while statement prematurely, the break statement can be used. ‘continue’ jumps back to the top of the loop.

**For loop –**

Iterating through a list using while loop requires a lot of code. So python provides for loop.

**Functions –**

To reuse the code again and again, functions are created. The code block within every function starts with a colon and is indented. Functions must be defined before they are called.

* Function arguments are defined within parentheses.
* Function arguments can be used as variables inside the function definition. However they can’t be referenced outside the function definition.
* In order to return any value from function, ‘return’ keyword is used also ‘return’ can’t be used outside function definition.
* Once you return a value from a function, it immediately stops executing. Any code after return statement will never get executed.
* Functions can be treated as normal objects means they can be assigned and reassigned to variables and later referenced by those names. Functions can also be used as arguments of other functions.

**Modules –**

Modules are pieces of code that other people have written to fulfill common tasks.

* The basic way to use module is to add ‘import module\_name’ at the top of your code and then using ‘module\_name.var’ to access functions and values with the name ‘var’ in the module

import random

for i in range (5):

value=random.randint(1,6)

print (value)

* If user need only certain function from the module then it take the form as ‘from module\_name import var’ and then var can be used as it is defined normally in the code.

from math import pi

print (pi)

* User can import a module or object under a different name using ‘as’ keyword. This is mainly used when module or object has a long or confusing name.

from math import sqrt as square\_root

print (square\_root(100))

**Classes -**

* Functions within class are called ‘methods’.
* Data within class is called ‘attributes’.
* If user wants to call method on object, he has to put brackets at end (as it’s a function) whereas it is not required while accessing attributes of a class.
* ‘Object\_1.method\_1()’ is exactly same as ‘Class\_1.method\_1(object\_1)’.
* ‘Class Variables’ are those which are shared among all instances of a class.
* Regular methods accept ‘self’ (object instance) as their first argument.
* Class methods accept ‘cls’ (class) as their first argument.
* Static methods do not accept any default argument.

**Arithmetic Operations**

1. Exponentiation (\*\*)

2\*\*5 🡪 32

1. Quotient (// floor division)

20//6 🡪 3

1. Remainder (% modulo operator)

1.25%0.5 🡪 0.25

**Data Types**

1. Lists 🡪 [] are mutable (i.e. can be changed)
2. Tuples 🡪 () are immutable
3. Dictionaries 🡪 {:}
4. Sets 🡪 {}

**List Slicing**

Squares=[0,1,4,9,16,25,36,49,64,81]

Squares[2:6] = [4,9,16,25]

Squares[:4] = [0,1,4,9]

Squares[7:] = [49,64,81]

Squares[::-1] = [81,64,49,36,25,16,9,4,1,0]

**Range Function**

a=list(range(5))

print (a)

Output 🡪 [0,1,2,3,4]

b=list(range(2,5))

print (b)

Output 🡪 [2,3,4]

General –

* Python exposes everything that exists in an object when you use the ‘dir’ function.
* \_\_doc\_\_ method is used to find the info about any object. In case of class, it returns whatever is written in docstrings.