BSc Computer Science – Allied Practical Python

Prepared by

Dr.K.Pazhanikumar

Head., Dept of Computer Science

S.T.Hindu College

Nagercoil

Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

TOP 5 PROGRAMMING LANGUAGES FOR SERVER-SIDE WEB DEVELOPMENT

- Python
- Ruby
- Node.js
- PHP
- Go

PYTHON

Python is the most interesting and profitable programming language to learn in the current scenario. It is not just a server-side programming language, a complicated application can be developed using it. The syntaxes are programmer-friendly, especially for the beginners. It comes with a large library with so many different pre-coded functions.

Execute Python Syntax

Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile.py

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!") The number of spaces is up to you as a programmer, but it has to be at least
```

Use the same number of spaces in the same block of code, otherwise Python will give you an error one.

Python Variables

In Python variables are created the moment you assign a value to it:

Example

Variables in Python:

```
x = 5
y = "Hello, World!“
```

Python has no command for declaring a variable.

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Example

Comments in Python:

```
#This is a comment.
print("Hello, World!")
```

Python does not really have a syntax for multi line comments.

To add a multiline comment you could insert a # for each line

Python Data Types

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type: str Numeric Types:int, float,

Complex Sequence Types: list, tuple, range

Mapping Type:dict

Set Types:set, frozenset

Boolean Type:bool

Binary Types:bytes, bytearray, memoryview

Getting the Data Type

The data type of any object by using the type() function:

Example

Print the data type of the variable x:

```
x = 5
print(type(x))
```

Python Numbers

There are three numeric types in Python:

- int
- float
- complex

Variables of numeric types are created when you assign a value to them:

Example

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
```

To verify the type of any object in Python, use the type() function:

Example:

```
print(type(x))
```

Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example

Integers:

```
x = 1
y = 35656222554887711
z = -3255522
```

Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Example

Floats:

x = 1.10
y = 1.0
z = -35.59

Float can also be scientific numbers with an "e" to indicate the power of 10.

Complex

Complex numbers are written with a "j" as the imaginary part:

Example

Complex:

x = 3+5j

y = 5j

z = -5j

Type Conversion

Convert from one type to another with the int(), float(), and complex() methods:

Example

Convert from one type to another:

```
x = 1 # int
    y = 2.8 # float
    z = 1j # complex

    #convert from int to float:
    a = float(x)

    #convert from float to int:
    b = int(y)

    #convert from int to complex:
    c = complex(x)
```

Python Input, Output and Import

Two built-in functions print() and input() to perform I/O task in Python.

print()

print('This sentence is output to the screen

a = 5

print('The value of a is', a)

print(1,2,3,4)# Output: 1 2 3 4

print(1,2,3,4,sep='*‘)#Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')#Output: 1#2#3#4&

We can even format strings like the old sprintf() style used in C programming language. We use the % operator to accomplish this.

```
>>> x = 12.3456789
>>> print('The value of x is %3.2f' %x)
The value of x is 12.35
>>> print('The value of x is %3.4f' %x)
The value of x is 12.3457
```

Input

To allow flexibility we might want to take the input from the user. In Python, we have the input() function to allow this. The syntax for input() is

input([prompt])

num = input('Enter a number: ')

Enter a number: 10

Import

When our program grows bigger, it is a good idea to break it into different modules.

A module is a file containing Python definitions and statements. Python modules have a filename and end with the extension .py.

Definitions inside a module can be imported to another module or the interactive interpreter in Python. We use the import keyword to do this.

For example, we can import the math module by typing in import math.

```
import math

print(math.pi)
```

Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

int() - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)

float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)

str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Example

```
x = int(1)   # x will be 1
y = float(2.8)   # y will be 2.8
x = str("s1") # x will be 's1'
```

Python Strings

String Literals

String literals in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function:

Example

```
print("Hello")
print('Hello')
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

Multiline Strings

Assign a multiline string to a variable by using three quotes:

Example

You can use three double quotes:

```
a = """Lorem ipsum dolor sit amet,
    consectetur adipiscing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua."""
    print(a)
```

Or three single quotes:

Example

```
a = '''Lorem ipsum dolor sit amet,
    consectetur adipiscing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.'''
    print(a)
```

In the result, the line breaks are inserted at the same position as in the code.

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

Example

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

Slicing

Return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Conditional Statement(If)

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Short Hand If

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

Short Hand If ... Else

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

Looping Statements-while

```
 i = 1
while i < 6:
  print(i)
  i += 1
```

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Looping Statements-for

- 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.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)

list = ["geeks", "for", "geeks"]
for index in range(len(list)):
    print list[index]
```

Arrays

Arrays are used to store multiple values in one single variable:.

Create an array containing car names:

```
cars = ["Ford", "Volvo", "BMW"]
```

Use the len() method to return the length of an array

Print each item in the cars array:

```
for x in cars:
    print(x)
```

Functions

In Python a function is defined using the def keyword:

```
def my_function():
    print("Hello from a function")
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
    print("Hello from a function")

my_function()
```

Function(Arguments)

```
def my_function(fname):
      print(fname + " Refsnes")

    my_function("Emil")
    my_function("Tobias")
    my_function("Linus")
```

Program 1

Write a menu driven program to convert the given temperature from Fahrenheit to Celsius and vice versa depending upon user‟s choice.

```
def menu():
        print("\n 1.Fahrenheit to Celsius")
        print("\n 2.Celsius to Fahrenheit")
        print("\n 3.Exit")
        choice=int(input("\n Enter Your Choice:"))
        return choice
def ftoc(f):
        c=(f-32)/1.8
        return c
def ctof(cen):
        far=1.8*cen+32
        return far
```

```
def main():
        ch=menu()
        if ch==1:
                f=float(input("\n Enter the Fahrenheit
Value:"))
                cel=ftoc(f)
                print("\n The Celisus Value is:%f"%(cel))
        elif ch==2:
                cen=float(input("\n Enter the Celsius
Value:"))
                far=ctof(cen)
                print("\n The Fahrenheit Value
is:%f"%(far))
        else:
                print("\n Invalid")
main()
```

Program 2

Write a menu-driven program, using user-defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.

```
def menu():
    print("\n 1.Area of the Circle")
    print("\n 2.Area of the Square")
    print("\n 3.Area of the Triangle")
    print("\n 4.Area of the Rectangle")
    print("\n 5.Exit")
    choice=int(input("\n Enter Your Choice:"))
    return choice
def circle(r):
    c=3.14*r*r
    return c
def square(a):
    s=a*a
    return s
```

```
 def triangle(b,h):
t=0.5*b*h
        return t
def rectangle(l,w):
        r=l*w
        return r
def main():
        ch=menu()
        if ch==1:
                print("\n Area of Circle Calculation")
                r=float(input("\n Enter the Radius Value:"))
                cir=circle(r)
                print("\n The Area of the Cricle is:%f"%(cir))
        elif ch==2:
                print("\n Area of Square Calculation")
                a=float(input("\n Enter the Side of the Square:"))
                sq=square(a)
                print("\n The Area of the Square is:%f"%(sq))
```

```
elif ch==3:
            print("\n Area of Triangle Calculation")
            b=float(input("\n Enter the Base of the Triangle:"))
            h=float(input("\n Enter the Height of the Triangle:"))
            tri=triangle(b,h)
            print("\n The Area of the Triangle is:%f"%(tri))
      elif ch==4:
            print("\n Area of Rectangle Calculation")
            l=float(input("\n Enter the Length of the Rectangle:"))
            w=float(input("\n Enter the Width of the Rectangle:"))
            tri=triangle(b,h)
            print("\n The Area of the Triangle is:%f"%(tri))
      else:
            print("\n Invalid Choice")
            ch=menu()
main()
```

Program 3

Write a program to display the first n terms of Fibonacci series

```
print("Fibonacci Series")
n=int(input("\n Enter the 'N' Value:"))
a=0
b=1
i=1
print("\n %d"%(a))
print("\n %d"%(b))
while i<n:
        c=a+b
        print("\n %d"%(c))
        a=b
        b=c
        i=i+1
```

Program 4

Write a program to find factorial of the given number.

```
print("Factorial of a Given Number")
n=int(input("Enter the Given Number:"))
s=1
i=1
while i<=n:
        s=s*i
        i=i+1
print("Factorial of Given Number is:%d"%(s))
```

Program 5

Write a program to find sum of the following series for n terms: 1 – 2/2! + 3/3! - - - - - n/n!

```
n=int(input("Enter No. of Terms:"))
def fact():
        s=0.
        f=1.
        i=1.
        while i<=n:
                f=f*i
                s=-s+(i/f)
                i+=1
        return s
def main():
        print("Sum of Series 1-2/2!+3/3!...n")
        sum=fact()
        print("Sum of Series=%f"%(sum))
main()
```

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.

Program 6

Write a program to calculate the sum of two compatible matrices.

```
m1=[[1,2,3],[4,5,6]]
m2=[[3,5,2],[1,3,4]]
m3=[[0,0,0],[0,0,0]]
m=len(m1)
n=len(m1[0])
o=len(m2)
p=len(m2[0])
if m==o and n==p:
        print("\n 'A' Matrix")
        for r in m1:
                print(r)
        print("\n 'B' Matrix")
        for r in m2:
                print(r)
        for i in range(m):
                for j in range(p):
                        m3[i][j]=m1[i][j]+m2[i][j]
        print("\n Answer Matrix")
        for r in m3:
                print(r)
```

Program 7

Write a program to calculate the product of two compatible matrices.

```
m1=[[0,0,0],[0,0,0],[0,0,0]]
m2=[[0,0,0],[0,0,0],[0,0,0]]
m3=[[0,0,0],[0,0,0],[0,0,0]]
m=len(m1)
n=len(m1[0])
o=len(m2)
p=len(m2[0])
if  n==o:
        for i in range(m):
                for j in range(n):
                        m1[i][j]=int(input("Enter 'A' Matrix:"))
        for i in range(o):
                for j in range(p):
                        m2[i][j]=int(input("Enter 'B' Matrix:"))
        for i in range(m):
```

```
for j in range(p):
                    for k in range(n):
                        m3[i][j]=m3[i][j]+m1[i][k]*m2[k][j]
    print("\n 'A' Matrix")
    for r in m1:
        print(r)
    print("\n 'B' Matrix")
    for r in m2:
        print(r)
    print("\n Answer Matrix")
    for r in m3:
        print(r)
```

Program 8
Write a program to explore String functions.

```
str1='Python String'
print("Print the Frist Character of the String")
print(str1)
print(str1[0])
print("Print the Last Character of the String")
print(str1)
print(str1[-1])
print("Print the Last Two Characters of the String")
print(str1)
print(str1[-2:])
```

```
print("Print the Characters from the Seventh Character of the
    String")
print(str1)
print(str1[7:])
print("Print the Characters till the Sixth Character of the String")
print(str1)
print(str1[:6])
str2=str1.islower()
print("Check all the Characters in the String are in Lower Case")
print(str1)
print(str2)
str3=str1.isupper()
print("Check all the Characters in the String are in Upper Case")
print(str1)
print(str3)
a=len(str1)
print(str1)
print("Length of Given String str1 is:",a)
```

```
str4=str1.capitalize()
print("Captialize the Characters of the String")
print(str1)
print(str4)
str5=str1.upper()
print("Convert all the Characters into Uppercase")
print(str1)
print(str5)
str6=str1.lower()
print("Convert all the Characters into Lowercase")
print(str1)
print(str6)
b='Hello'
c='World'
d=b+c
print("String Concentration")
print(d)
```

CSV File

CSV literally stands for comma separated variable, where the comma is what is known as a "delimiter." While you can also just simply use Python's split() function, to separate lines and data within each line, the CSV module can also be used to make things easy.

Example CSV file data:

1/2/2014,5,8,red

1/3/2014,5,2,green

1/4/2014,9,1,blue

Program 9
Creating a CSV File based on user input.

```
import csv
with open('salary1.csv','w')as fp:
    a=csv.writer(fp,delimiter=',')
    for i in range(1,5):
            d1=input("Employee id:")
            d2=input("Salary:")
            data=[(d1,d2)]
            a.writerows(data)
fp.close()
```

Program 10

. Reading a CSV File already created and display the contents

```
import csv
with open('score.csv','w')as fp:
    a=csv.writer(fp,delimiter=',')
    data=[['Name','Age'],['Vishak','19'],['Godha','20'],['Vedha','21']]
    a.writerows(data)
fp.close()
print("CSV File created")
print("Content of CSV File")
with open('score.csv','r')as scorefile:
    scorefileReader=csv.reader(scorefile)
    scorelist=[]
    for row in scorefileReader:
        if len(row)!=0:
            scorelist=row
            print(scorelist)
scorefile.close()
```