Python Programming Basics

By
Vishwas K Singh

Version : 1
Date : 07 Sept 2017

For any feedbacks and improvements you can contact the author

Vishwas K Singh
email : vishwasks32@gmail.com

For any queries contact the author or visit www.python.org

Python Basics

Basics of programming:

How do you approach a programming problem?

Think of a program as a black box which transforms an input to a specified output.

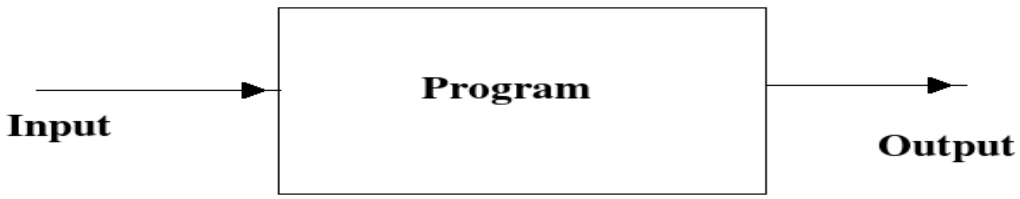

Fig 1

Finding the logical solutions to the problems in a language a computer understands means programming.

Types of Programming languages:

Compiled Languages : C, C++...
Interpreted Languages : Python, Java, JavaScript …

Processes steps :

Compilation :

Source Code → compile → Binary Code → execute

Interpretation :

Source Code → interpret(execute)
Source Code → Byte Code → interpret

Python Intro :

- Python is an ***interpreted*** language similar to Java
- It needs an ***interpreter*** to run
- It was invented by ***Guido Van Rossum*** in 1991
- Current version of Python is ***3.6.2***
- You can also find codes of ***2.7.5*** or above
- You need to install python Interpreter before you can execute python programs
- Interpretation means executing commands line by line
- Here we can execute in two modes
    1. Interactive mode
    2. Script mode
- The motto of such languages in common is
  "Write Once Run Anywhere"
- The interpreter for different OS is different but your python code is same
- The usual interpreter is written in ***C*** and ***Python*** called ***Cython, PyPi***
- One written in ***Java*** is called ***Jython***
- You can also write java code in python and implement in jython
- to start the interpreter
  start → cmd → type python
- '>>>' represents a prompt, here you can execute python commands in interactive mode
- Try the examples and observe output :
  2+3
  2/2
  '123' + '456'
  'Hi'+'Dear'
  25/0
- Everything in (' ', **" ",** ''' ''') denotes strings

- Note : The '+' operator, if supplied with **two numbers adds** them, if **strings** then **concatenates** them
- also observe **divide by zero** error

- To Create a script you can use notepad, idle(comes with python), or any of your favorite text editor

Exercise :

1. Search the internet for some python programs
2. Think of a program that you want on your computer

Getting Help:

- Some times we tend to forget the modules and function names
- to get help in the interpreter about a module

```
import mod_name
dir(mod_name)
help(funct_name)
```

Basics of Language : (.py extension for python source code)

Example 1: Create a script ex1.py

```
print('Hello world')
```

Save the file and to execute

```
python ex1.py
```

- So you did the first program, to output anything to the console we use print()
- Everything you do on a computer is data manipulation, how do you access data in memory ?
- Memory can be accessed through locations, to address the locations we use variables
- variables means name to locations
- variable names can be alphanumeric with special characters _
- '=' denotes **assignment** operator
- **+=,-=,*=** denote **augmented assignment**, means operation is performed with previous value of variable with that in statement
- python is ***case sensitive***
- Note:
  1. dont start variabe names with numbers
  2. Try to self document the variables, so that when you read your code after some time, you will be able to recognize it.
  3. Comments start with “#” in python, can be used to comment blocks of code

- How do you take the input ?

Example 2: filename ex2.py

```
my_name = str(input('Enter Your Name : '))
print('Hi ' + my_name)
#print('Hi %s'%(my_name))
```

- str() converts the input to a string
- there are two print statements which show two ways of using print variables

Exercise:

1. Write a script to add two numbers

Data Types in python:

- Data can be represented in python as a number which may be an
  1. integer → signed
  2. float → floating point real values
  3. long → long integers which can be hexadecimal or octal too
  4. complex

- Other data types are list, tuples, dictionary, objects
- You dont need to define previously the type of a variable before using it(dynamically typed), the interpreter does that for you when you assign some value to it.

Operators in Python:

- +,-,*,/,%, ** are arithmetic operators
- and, or, not are logical operators
- <, >, <=, >=, == ,!=,<> are conditional operators

Strings:

- Strings in python are continuous set of characters in between quotation marks
- Subsets of strings can be accessed by slice operators [] and [:]
- Indexes starting with 0 at beginning of the string
- and -1 at the end
- + and * can be used to concatenate and repetition

Example 3: ex3.py

```
nme = 'vishwas'
nme1 = 'king'
print(nme[1])
print(nme[2:4])
print(nme[2:])
print(nme[:6])
print(nme1+' '+nme[4:7])

k = ''
for i in range(len(nme)-1,-1,-1):
        k+=nme[i]
print(k)
print('-'*20)
print(nme*2)
```

- execute the above program and observe the outputs

Lists and Tuples:

- Lists can be thought of as arrays or continuous memory locations which can be accessed by starting locations
- Lists can store various types of data in them
- size and types of items in a list can be changed
- They are accessed through index offsets
- Indexes start at 0
- List can be accessed similar to strings
- Tuples are, we can say as read-only lists(immutable)

Example 4: ex4.py

```
aList = [] # an empty list
bList = [1,2,3]
print(bList[2])
bList.append(4)
print(bList.pop())
print(bList)
bList.remove(2)
print(bList)
```

```
# Tuples
ktuple = (1,2,3,4)
print(ktuple)
print(len(ktuple))
print(ktuple[2])
print(ktuple[2:4])
```

- Note : here tuples and lists can contain different types of data

Dictionaries:

- Dictionaries are use to store data such as key value pairs
- keys can be any data types but usually are ***Strings or Integers***
- values can be any python types

Example 5: ex5.py

```
dDict = {}
dDict['name'] = 'vishwas'
dDict['USN'] = '4VZ16LDS23'
print(dDict)
print(dDict['USN'])
print(dDict.keys())
```

Exercise:

1. Write a program to convert between celcius to farenheit
2. write a program to reverse a given string

Code Blocks and Indentation:

- Code Blocks are formed in python by use of indentation
- indentation means alignment or spacing
- any block of code having same alignment are treated to be inside the block
- No use of curly braces or any other special characters for creating blocks

Conditionals and loops:

```
if expression:
        stmt1
        stmt2
if expression:
        stmt block
else:
        stmt block


if expression:
        stmt block
elif condn:
        stmt block
else:
        stmt block
```

- the above three conditional statements are shown
- the else – if block in python is elif
- the expression or condition is a non zero integer or boolean value True
- if its so then statement block inside is executed, else not

Exercise:

1. write a program to check for a given string to be palindrome
2. write a program to check the given number is even or odd
3. write a program to check for a perfect square or not
4. write a program to print prime numbers below 100

Project:

Write a number to words converter, using if and elifs and dicts between 1 to 10

While Loops:

```
while condn:
    statements
```

- The above shows the while loop which executes until the condn is false
- in place of condn we can use boolean values **True or False**

For loop and range():

```
for i in range():
        statements
```

- The for loops can be iterated over a list of numbers, indexes or any list of items
- range(n) generates a list from 0 to n-1

Example 6: ex6.py

```
print('Display usage of for')
for i in range(10):
        print(i)

kname = 'vishwas'
for i in range(len(kname)):
        print(i," ", k[i])



for i in [1,2,3,4,5]:
        print(i)
```

File Handling in python:

- file can be opened, read, written and closed
- the file name is expected to be present in the current working directory
- if not **absolute paths** must be provided

Example 7: ex7.py

```
# to open a file
hndle = open("file_name", access_mode='r')
# Read all contents of file to dat1
dat1 = hndle.read()

for eachLine in dat1:
        print(eachLine)

hndle.close()
```

- file_name is the file to be opened and either can be opened in 'r' – read mode, 'w' – writemode or 'w+' - append mode
- the default access mode is a read mode
- 'b' opens the file is a binary mode, very useful for reading binary files

Errors and Exceptions:

- Python can check for compilation errors and also it can raise exception
- The developer can check for runtime errors or exception and write code to handle such exceptions by wraping the code in a try-except block

```
try:
        stmts
except someError:
        stmts
```

Functions:

```
def functionName([arguments]):
        block of stmts
        return values

#  To call a functions
functionName([parameters])
```

- the functions are similar to as in any language
- functions are used when we have to repeat some set of statements
- ***def*** is used to define functions, they can return some values or may be not, if they don't return any thing the ***None*** object is returned

Example 8:ex8.py

```
# A calculator Script
def add_num(a,b):
        return a+b
def sub_num(a,b):
        return a-b
```

```
def mul_num(a,b):
        return a*b
def div_num(a,b):
        return a/b

if __name__ == '__main__':
        print('Welcome to calculator')
        print('Menu:')
        print('1.add')
        print(2.Subtract)
        print(3.Multiply)
        print(4.Divide)

        while True:
                num1 = int('Enter the first number: ')
                num2 = int('Enter the second number: ')
                chice = int('Enter the choice: ')
                if chice in [1,2,3,4]:
                        if chice == 1:
                                print(add_num(num1,num2))
                        elif chice == 2:
                                print(sub_num(num1,num2))
                        elif chice == 3:
                                print(mul_num(num1,num2))
                        elif chice == 4:
                                if num2 != 0:
                                        print(div_num(num1,num2))
                                else:
                                        print('Cannot divide by zero')

                else:
                        print('Invalid choice!!')
```

Module:

- modules in python are a way to physically organize and distinguish related pieces of python code into individual files
- modules can contain executable code, functions, classes etc...
- name of a module is the filename without '.py' extension
- to use the module ***import module_name*** is used

- to access a variable or a function in a module
  module_name.function()
  module_name.variable

Classes:

- Classes are just like containers of data can also be called as user defined datatypes
- they contain attributes and methods to access and update those attributes

```
class class_name([any_base_class_extensions]):
    '''optional document string'''
    static_member_declarations
    member_declarations
```

Example 9: ex9.py

```
class user:
    ''' A class for user in address book '''
    version = 0.1
    def __init__(self, nm):
        'A constructor'
        self.name = nm
    def getname(self):
        'returns the name of instance'
        return self.name
    def getver(self):
        'returns version'
        return self.version


if __name__ == '__main__':
    # Creates an instance of the class described above
    f1 = user('vishwas')
    print(f1.getname())
```

Note: use of some keywords are omitted in this notes ***continue, break, pass***

Iterator and generators:

- Iterable objects means like a list of values which can be visited one-by-one
- for i in range(n): is one such function which uses for statement to iterate over the objects of the list.
- the iter() function can be used to an iterable object such as a list,set,dict,file
- Generators are functions like

```
yield expression

def primes():
    yield 2
    yield 3
    yield 5
    yield 7
    return
```

- They return a set of values

list Comprehensions:

- It is an expression that combines a function, for statement and an optional if statement

Example 10: ex10.py

```
# Generate all even numbers under 22
print([2 * x for x in range(22)])

# print a list of two tuples
print([(x,x) for x in (2,3,4,5)])

# print a list of 10 random numbers
import random
print([random.random() for x in range(10)])
```

Lambda functions:

- It is similar to defined functions with parameters
- but it is only a single expression

```
lambda a: a[0]*2 + a[1]
lambda x: x* a
```

Projects:

1. A Number to word converter which handles upto 20 numbers
2. A Number to word converter which handles any number of inputs including negatives
3. An address book application to add, view, delete, store addresses of people
4. An mp3 id3 tag reader, read an mp3 file and display its details using headers
5. A board game in python, where a pawn is hidden in an array, a users guesses its position, if the guess is correct the user wins or else he is given another chance
6. write a python source code reader in python, extend it to check for different source codes too

   (advanced)
7. write a webpage scraper which gets all the links in a web page
8. write a simple web server client in python(use sockets)