Python Programming

Dr. Shailesh Sivan

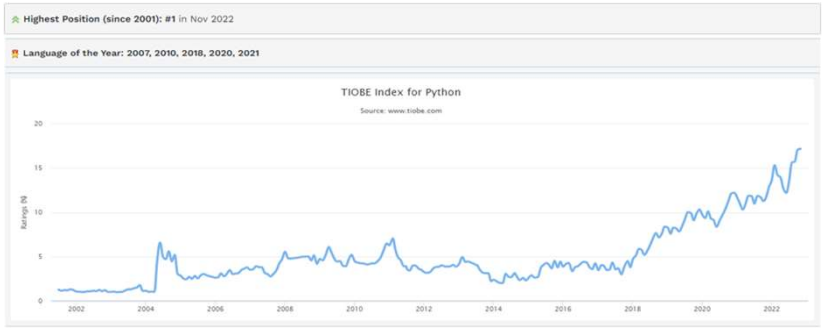
1

Top Programming Languages in 2022-Python is still No. 1 .

IEEE Spectrum - an award-winning technology magazine and the flagship publication of the IEEE.

2

TIOBE Index



*The TIOBE Programming Community index is **an indicator of the popularity of programming languages.** The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors*

3

Overview

- Introduction to Python
- Python Development Tools
  - Installation
  - Package Managers and Virtual Environments
- Language Fundamentals
  - Basic Data Types
  - Operators
  - String Operations
  - Control Structures
  - Containers
  - Functions
  - Classes

4

Python Introduction

- **Python** is a widely used **general-purpose**, **high level programming language.**
- Initially designed by **Guido van Rossum** in **1991** and developed by **Python Software Foundation**.
- There are 2 major Python versions : **Python 2** and **Python 3**

- Three main popular **applications** for Python:
  1. Web Development
  2. Data Science-including machine learning, data analysis, and data visualization

5

Python Development Tools

Installation

- **Windows** - Download and Install the Python 3 Installer (https://www.python.org/downloads/windows/)
- **Linux(ubuntu)** - There is a very good chance your Linux distribution has Python installed already; otherwise

  $ sudo apt-get update

  $ sudo apt-get install python3.X
- For **AI/ML**- Install
  - **Anaconda**[about 3 GB to install over 720+ packages (many of the packages are never used) (https://www.anaconda.com/distribution/)
  - **Miniconda**[Know what package(s) you need to install] (https://docs.conda.io/en/latest/miniconda.html)

6

Python Development Tools

- **Python Package Managers** - Python utility intended to simplify the tasks of locating, installing, upgrading and removing Python packages.
  - **pip - pip** is a package management system used to install and manage software packages along with its dependencies

    $pip install < you package >
  - **conda** - is an open source package management system and environment management system that runs on Windows, macOS and Linux

    $conda install < you package >

7

Python Fundamentals

Start with 'Hello World'

```
print('Hello World')
# Hello World
```


8

Google Colab

9

Python Fundamentals

Basic Data Types

```
a=5
print(a)
print(type(a))
# 5
# <class 'int'>

f=1.5
print(f)
print(type(f))
# 1.5
# <class 'float'>

s="hello"
print(s)
print(type(s))
# hello
# <class 'str'>
```

```
b=True
print(b)
print(type(b))
# True
# <class 'bool'>

t=5+3j
print(t)
print(type(t))
# (5+3j)
# <class 'complex'>
```


10

Python Fundamentals

Arithmetic Operators

```
a = 7
b = 3

ab_sum = a + b
print(ab_sum)
# 10

ab_dif = a - b
print(ab_dif)
# 4

ab_pro = a * b
print(ab_pro)
# 21
```

```
ab_quo = a / b
print(ab_quo)
# 2.3333333333333335

ab_iquo = a // b
print(ab_iquo)
# 2

ab_rem = a % b
print(ab_rem)
# 1

ab_pow = a ** b
print(ab_rem)
# 343
```


11

Python Fundamentals

Boolean Operations

```
T=True
F=False

print(T,F)
# True False

p = 5 > 3
print(p)
# True

q = -1 < -12.5
print(q)
# False
```

Note

Other relational opeators <=, >=, ==, !=

```
print(p and q)
# False

print(p or q)
# True

print(not q)
# True
```

Note

Python uses words instead of symbols like &&, ||, ! for Boolean

12

Python Fundamentals

Some String Operations

```
s = 'hello'
u = "hello"

print(s)
print(u)
# hello
# hello

s1 = "python"
s2 = 'world'
s3 = s1 + ' ' + s2
print(s3)
# python world

s3 = '%s %s %d' %(s1, s2, 1011)
print(s3)
# python world 1011

print(len(s3))
# 12
```

```
print(s3.upper())
# PYTHON WORLD

print(s3.capitalize())
# Python world

print(s3.lower())
# python world

print('hello world how are you'.split(' '))
# ['hello', 'world', 'how', 'are', 'you']

print('book'.replace('o','e'))
# beek

word = 'jewellery'
print(word.find('well'))
print(word.find('is'))
# 2
# -1
```


13

Control Structures

if - else

```
if number > 99 and number < 1000:
    print('3 digit')
else:
    print('Not 3 digit')

# Enter number : 123
# 3 digit
```

Note

- Take care of indentation !
- Don't forget to put ':' at the end
- Remember its **elif** not **else if**

if – elif - else

```
response = input('Are you familiar with
python : ')

if response.upper() == "YES":
    print("You can skip this course :-|")
elif response.upper() == "NO":
    print("You are at the right place :-)")
else:
    print('Sorry wrong input :-(')

# Are you familiar with python : no
# You are at the right place :-)
```


14

Control Structures

for loop

```
for x in range(10):
    print(x,end=' ')

# 0 1 2 3 4 5 6 7 8 9


limit = int(input('Enter a limit : '))
sum = 0

for i in range(1,limit + 1):
        if i%2 != 0:
                sum += i

print("Odd sum = "+str(sum))

# Enter a limit : 15
# Odd sum = 64
```

```
range()

print(list(range(10)))
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(list(range(1,10)))
# [1, 2, 3, 4, 5, 6, 7, 8, 9]

print(list(range(1,10,2)))
# [1, 3, 5, 7, 9]
```


15

Control Structures

While loop

```
number = int(input('Enter number : '))
s = 0
while number > 0 :
    s += number%10
    number = number//10
print(s)

# Enter number : 1254
# 12
```

Nested loops

```
limit = int(input('Enter number : '))
for num in range(2,limit+1):
    is_divisible = False
    k=2
    while k <= num//2 :
        if num % k == 0:
            is_divisible=True
            break;
        k += 1
    if not is_divisible:
        print(num,end=' ')

# Enter number : 400
# 2 3 5 7 11 13 17 19 23 29 31 37
```


16

Containers

Containers - List

```
mylist = ['a','b', 1, 1.2, True]
print(mylist)
# ['a', 'b', 1, 1.2, True]

mylist.append('new')
print(mylist)
# ['a', 'b', 1, 1.2, True, 'new']

print(mylist.pop())
# new

mylist.insert(2,'new')
print(mylist)
# ['a', 'b', 'new', 1, 1.2, True]

mylist.remove('new')
print(mylist)
# ['a', 'b', 1, 1.2, True]
```

```
b = [1,2,3]
mylist.append(b)
print(mylist)
# ['a', 'b', 1, 1.2, True, [1, 2, 3]]

mylist.remove(b)
print(mylist)
# ['a', 'b', 1, 1.2, True]

mylist.extend(b)
print(mylist)
# ['a', 'b', 1, 1.2, True, 1, 2, 3]

a = [2,3,1,4,5]
a.sort()
print(a)
# [1, 2, 3, 4, 5]

print(list('hello'))
# ['h', 'e', 'l', 'l', 'o']
```


17

Containers

Containers – List Slicing

```
print(numbers[1],numbers[-1])
# 2 10

sliced = numbers[5:11]
print(sliced)
# [6, 7, 8, 9, 10]

sliced = numbers[5:]
print(sliced)
# [6, 7, 8, 9, 10]

sliced = numbers[:7]
print(sliced)
# [1, 2, 3, 4, 5, 6, 7]

sliced = numbers[-2:]
print(sliced)
# [9, 10]
```

Note
list_name[a:b] => where it slices out a subset from the index a to b-1

18

Containers

Containers – List comprehension

```
numbers = list(range(1, 8))
print(numbers)
# [1, 2, 3, 4, 5, 6, 7]

square = []
for i in numbers:
    square.append(pow(i,2))
print(square)
# [1, 4, 9, 16, 25, 36, 49]
```

```
##List Comprehension
square = [x**2 for x in numbers]
print(square)
# [1, 4, 9, 16, 25, 36, 49]

## List comprehension with a filter
odd_square =[x**2 for x in numbers if x%2 != 0]
print(odd_square)
# [1, 9, 25, 49]
```

```
A = [4,6,8,9]
AxA = [(a,b) for a in A for b in A if a!=b]
print(AxA)

# [(4, 6), (4, 8), (4, 9), (6, 4), (6, 8), (6, 9), (8, 4),
(8, 6), (8, 9), (9, 4), (9, 6), (9, 8)]
```


19

Containers

Containers – Dictionary

```
person = {'name' : 'Manu', 'age': 28}
print(person['name'])
# Manu

print('name' in person)
# True

print('sex' in person)
# False

person['sex'] = 'male'
print(person)
# {'name': 'Manu', 'age': 28, 'sex': 'male'}

for item in person:
    print(item,person[item])
# name Manu
# age 28
# sex male
```

```
for (key,value) in person.items():
    print(key.capitalize(),'\t:\t', value)
# Name    :       Manu
# Age     :       28
# Sex     :       male

print(person.keys())
# dict_keys(['name', 'age', 'sex'])
```

Note
A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

20

Containers

Containers - Tuples

```
t1 = (1,2,3)
t2 = 4,5,6
print(t1,t2)
# (1, 2, 3) (4, 5, 6)

t3 = t1 +t2
print(t3)
# (1, 2, 3, 4, 5, 6)

lt = tuple(['a','b','c','d'])
print(lt)
# ('a', 'b', 'c', 'd')

lt[2] = 'x'

# ----> 1 lt[2] = 'x'
# TypeError: 'tuple' object does not support
item assignment
```

Tuple with single element

```
s = (3)
print(type(s))
#<class 'int'>

s = (3,)
print(type(s))
#<class 'tuple'>
```


21

Containers

Containers – Sets

```
s = {1,2,3}
print(s,type(s))
# {1, 2, 3} <class 'set'>

fset = {"apple", "banana", "cherry"}
fset.remove("banana")
print(fset)
# {'apple', 'cherry'}

fset = {"apple", "banana", "cherry"}
fset.discard("banana")
print(fset)
# {'apple', 'cherry'}

fset = {"apple", "banana", "cherry"}
fset.clear()
print(fset)
# set()
```

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
# {1, 2, 3, 'b', 'c', 'a'}

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)
print(z)
# {'apple'}

lst = [1,2,3,4,5,5,5,7,6]
myset = list(set(lst))
print(myset)
# [1, 2, 3, 4, 5, 6, 7]
```


22

Functions

Functions

```
def twice(number):
    return 2*number
t = twice(5)
print(t)
# 10


def isPrime(number):
    for factor in range(2, (number//2)+1):
        if number%factor == 0:
            return False
    return True

number = int(input('Enter the number '))
print(isPrime(number))
# Enter the number 10
# False
```


23

Functions

Functions

```
def printPrimes(llimit, ulimit):
    for num in range(llimit, ulimit + 1):
        if isPrime(num) == True:
            print(num, end = ' ')

printPrimes(5,50)
# 5 7 11 13 17 19 23 29 31 37 41 43 47

def swap(x,y):
    t = x
    x = y
    y = t
    return x,y

a=5
b=7
a,b = swap(a,b)
print(a,b)
#7 5
```

Note
Python supports more than one return values

Note
Python support function with default arguments

```
def calculatePayable(p, y = 1, r = 5):
    return p*(1 + r* y/100)

print(calculatePayable(1000))
# 1050.0
print(calculatePayable(1000, y =3))
# 1150.0
print(calculatePayable(1000, r = 10, y=3))
# 1300.0
print(calculatePayable(5000, r = 3))
# 5150.0
```


24

Classes

Classes

```
class Adder:

    #constructor
    def __init__(self):
        self.__x = 0
        self.__y = 0
    #setter
    def setValues(self, x, y):
        self.__x = x
        self.__y = y

    #self variable name can be anything
    def calculate(self1):
        self1.__sum = self1.__x + self1.__y

    #getter
    def getSum(self):
        return self.__sum
```

Note

In object-oriented programming, a class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

'_' symbol before the data member - protected member
'__' symbol before the data member - private member

```
adder =  Adder()
adder.setValues(5,4)
adder.calculate()
print(adder.getSum())
#9
```


25

Thank You

26