**Python tutorial:**

**Agenda:**

* Python introduction
* Who uses python
* Python features
* Operators in python
* Data Types in python
* Flow control
* Functions in python
* File Handling in python

**Python Introduction:**

* Python was created by Guido Rossum in 1989 and is very **easy to learn**.
* Python is **interpreted**, **object oriented**, **high level** programming language with dynamic semantics.

**Who uses python:**

* The popular youtube video sharing service is largely written in Python
* Google, Dropbox, Raspberry pi, Bittorrent, NASA, NSA and Netflix

**Python features:**

* **Simple and easy to learn:** python is a simple and easy to learn, read & write.

think below java code and python, length of python code is less.

python code:

for i in range(10):

print(i)

java code:

for(int i =0;i<10;i++)

{

System.out.println(i)

}

* **Free and open source:** Python is an example of FLOSS (Free/Libre and Open source Software) which means one can freely distribute copies of this software, read it’s source code, modify it, etc.
* **High-level language:** One does not need to bother about the low-level details like memory allocation, etc.while writing script.
* **Portable:** Supported by many platforms like Linux, Windows, Macintosh, etc.
* **Interpreted language:**

java is compile and interpreted programming language

javac Test.java

java Test

In python interpreter is going to take care compile and gives output

python Test.py

* **Supports different programming paradigm:** Python supports procedure-oriented as well as object-oriented programming.

without writing any class we able to write python programs.

>>> x = 10

>>> def f1():

... print("hello",x)

...

>>> f1()

('hello', 10)

* Extensible: Python code can invoke C and C++ libraries, can be called from and C++ programs, can integrate with Java and .Net components.
* Dynamically typed

we no need to specify data type

python code:

x = 10

print(x)

java code:

class Hello{

int x = 10

System.out.print(x)

}

>>> x = 10

>>> print(type(x))

<type 'int'>

>>> x = 10.0

>>> print(type(x))

<type 'float'>

>>> x = True

>>> print(type(x))

<type 'bool'>

**Extensive Library:**

>>> import math

>>> print(math.sqrt(4))

2.0

**Platform independent:**

Write once and run anywhere.

**Potability:**

migration from one system to another

**where we can use python:**

* We can develop desktop applications.
* We can develop web applications(django,flask,)
* Database applications
* It Automation
* For Networking applications
* Games
* Data analysis.
* Machine learning
* AI applications
* IOT applications.

**Limitations of Python**

* Performance
* Mobile applications

**Python Installation:**

* Go to python.org
* Download python based on your operating system.
* Verify python installation using below command in command prompt or terminal **python --version**

**Operators in Python:**

* **Arithmetic operators**

Addition(+)

Subtract(-)

Multiply(\*)

Divide(/)

Exponential(\*\*) example:2\*\*3

Modulus(%)

float division(//)

**examples:**

>>> num1 = 10

>>> num2 = 20

>>> print("num1 + num2 =",num1 + num2)

('num1 + num2 =', 30)

>>> print("num1 - num2 =",num1 - num2)

('num1 - num2 =', -10)

>>> print("num1 \* num2 =",num1 \* num2)

('num1 \* num2 =', 200)

>>> print("num1 / num2 =",num1 / num2)

('num1 / num2 =', 0)

>>> print("num1 % num2 =",num1 % num2)

('num1 % num2 =', 10)

>>> print("num1^3 =",num1\*\*3)

('num1^3 =', 1000)

* **Assignment operators**

+=

-=

\*=

/=

%=

//=

\*\*=

**example:**

>>> num3 = num1 + num2

>>> print(num3)

30

* **Comparison operators**

**examples:**

>>> print("Is num1 > num2 =",num1 > num2)

('Is num1 > num2 =', False)

>>> print("Is num1 < num2 =",num1 < num2)

('Is num1 < num2 =', True)

>>> print("Is num1 == num2 =",num1 == num2)

('Is num1 == num2 =', False)

>>> print("Is num1 != num2 =",num1 != num2)

('Is num1 != num2 =', True)

* **Logical operators**

and

or

not

**examples:**

>>> x = True

>>> y = False

>>> print("x and y =", x and y)

('x and y =', False)

>>> print("x or y =", x or y)

('x or y =', True)

>>> print("not of x =", not x)

('not of x =', False)

**Bitwise operators**

a | b (OR)

a & b (AND)

a ^ b (XOR)

a >> b

a << b

**examples:**

>>> num4 = 6 #110

>>> num5 = 2 #010

>>> print('Bitwise and = ',num4 & num5)

('Bitwise and = ', 2)

>>> print("Bitwise or = ", num4 | num5)

('Bitwise or = ', 6)

>>> print("Bitwise xor = ", num4 ^ num5)

('Bitwise xor = ', 4)

>>> print("num4 right shift by 2 ", num4 >> 2)

('num4 right shift by 2 ', 1)

>>> print("num4 left shift by 2 ", num4 << 2)

('num4 left shift by 2 ', 24)

**Identity Operators**

is

is not

**examples:**

>>> r = 10

>>> r is 10

True

>>> r is not 10

False

**Membership operators**

in

Not in

**examps:**

>>> s = [1,2,3,4,5]

>>> 3 in s

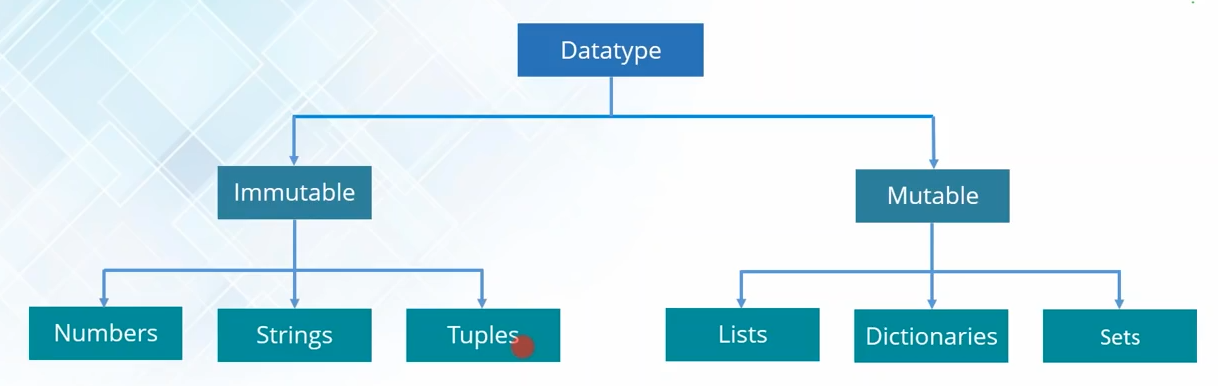
True

>>> 10 in s

False

**Data types in python:**

* Python is a loosely typed language. Therefore, no need to define the datatype of variables.
* No need to declare variables before using them.



**Numbers:** Integer, Float, Complex

Examples

integers:

1, 4 etc

4 +2 =6

2 \*\* 3 =8

float:

3.5,6.0 etc

2.0+5 = 7.0

6/3=2.0

compex

3+2j, etc

(3+4j)+(1+1j)=4+5j

(2+2j)\*2=4+4j

**Strings:**

Strings are sequences of one character string.

example:

sample = "Welcome to python tutorial"

or

sample = 'Welcome to python tutorial'

Multiline strings can be denoted using triple quotes '''or """

example:

sample = """Welcome to python tutorial this is how py works."""

operations in string:

Concatenation:

>>> string1 = 'Python'

>>> string2 = 'Tutorial'

>>> print(string1 + string2)

PythonTutorial

Repetition:

>>> print(string1 \* 2)

PythonPython

Slicing:

>>> print(string2[3:6])

ori

Indexing:

>>> print(string1[-1]+string1[3])

nh

Type specific operations:

find()

>>> string1.find('thon')

2.replace()

>>> str1 = 'Edureka'

>>> str1.replace('Ed','E')

'Eureka'

3.split()

string1 = 'Python'

>>> string1.split(',')

['Python']

>>> str = 'h,a,r,e,e,s,h'

>>> str.split(',')

['h', 'a', 'r', 'e', 'e', 's', 'h']

4.count()

>>> str2 = 'edureka'

>>> str.count('e',beg=0,end=6)

5.upper()

6.max()

7.min()

8.isalpha()

**Tuples:**

>>> my\_tup = ('hareesh','rajesh','anil','divya')

>>> print(my\_tup+('sandeep'))

concatenation:

>>> print(my\_tup+('sandeep','Nell'))

('hareesh', 'rajesh', 'anil', 'divya', 'sandeep', 'Nell')

repetition

>>> print(my\_tup\*3)

('hareesh', 'rajesh', 'anil', 'divya', 'hareesh', 'rajesh', 'anil', 'divya', 'hareesh', 'rajesh', 'anil', 'divya')

slicing

>>> print(my\_tup[1:3])

('rajesh', 'anil')

**Lists:**

Lists: a list is a sequence of mutable python objects like floating number,string literals etc

the lists can be modidifed.

tuples are defined using square braces.

concatenation

repetition

slicing

indexing

append(value)

extend(list)

insert(index,value)

pop()

lists examples:

>>> my\_colors =['voilet','indigo','blue','green','yellow','orange','red']

>>> my\_colors.append('grey')

>>> print(my\_colors)

['voilet', 'indigo', 'blue', 'green', 'yellow', 'orange', 'red', 'grey']

>>> my\_colors.extend(['white','black'])

>>> print(my\_colors)

['voilet', 'indigo', 'blue', 'green', 'yellow', 'orange', 'red', 'grey', 'white', 'black']

**Dictionary:**

empty dictionary:

myDict = {}

dictionary with integer keys

myDict={1:'apple',2:'orange'}

dictionary with mixed keys

myDict={'name':'john',1:[1,2,3,]}

from sequence having each item as a pair:

myDict=dict([(1,'apple'),(2,'ball')])

dictionary methods:

accessing dictionary

myDict={1:'apple',2:'orange'}

myDict[1]

'apple'

len():

len(myDict)

2

key():

myDict.key()

values():

myDict.values()

['apple', 'orange']

home-work:

items()

get()

update()

pop()

**Sets:**

A set is an unordered collection of items

every element is unique(no duplicates) and must be immutable (which cannot be changed)

set methods

creating set:

my\_set = {1,2,3}

union:

mys1 ={1,2,'a'}

mys2 ={1,'b','c'}

mys1 | mys2

op:{1,2,'a','b','c'}

intersection:

myS1={1,2,'c'}

myS2={1,'b','c'}

mys1 & mys2

op:{1,'c'}

difference:

myS1={1,2,'c'}

myS2={1,'b','c'}

mys1-mys2

Op:{2}

**Flow control statements:**

\*\*\*\*\*\*\*\*\*\*\*if\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*if..else..\*\*\*\*\*\*\*\*

syntax:

if(condition):

statements 1..

else:

statements 2..

\*\*\*\*\*\*\*\*if..elif...else\*\*\*\*\*\*\*\*

syntax:

if(condition 1):

statements 1..

elfi(condition 2):

statements 2..

else

statements 3..

marks = 75

if(marks > 80):

print("Grade A")

elif(marks > 60):

print("Grade B")

elif(marks >40) and (marks <= 60)

print("Grade C")

else:

print("Grade D")

\*\*\*\*\*\*\*\*\*\*\*for\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

repeat things till the given number of times

Syntax:

for iterator name in iterating sequence

execute statements.

for quant in range(99,0,-1):

\*\*\*\*\*\*\*\*\*\*\*while\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

repeat things until the loop condition is true

syntax:

while(condition is True):

num = int(input('Enter the value of n='))

if(num<=0):

print("Enter a valid value")

else:

sum=0

while(num>0):

sum+=num

num-=1

print(sum)

\*\*\*\*\*\*\*\*\*\*\*break\*\*\*\*\*\*\*\*\*\*\*\*

syntax

break

\*\*\*\*\*\*\*\*\*\*continue\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*functions in python\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A function is a block of prganized,reusable sets of instructions that is used to perform some related actions.

why do we use functions?

-re-usability of code minimizes redudancy

-procedural decomopostion makes things organized.

functions can be two types:

Built in function

User defined function

user defined function example:

syntax

def fun\_name(arg1,arg2,arg3,...)

statements...

return[expression]

example:

def add(a,b)

sum = a+b

return sum

num1 = 1

num2 = 2

print(add(num1,num2))

**Interview questions:**