**Basic python**

* **Introduction to python**
* **Python use cases**
* **The birth of python, python timeline, python vs other languages, featuresof python**
* **Versions of python, python distributions**
* **Data types:**

 a **data type** or simply type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data. Most programming languages support various types of data, for example: real, integer or Boolean.

**They are two types of data types**

1. **Static data types:**

Programmer should have to defined the datatype to the variable explicitly

1. **Dynamic data types:**

Programmer need not to define the data type to variable explicitly

**Python support following data types**

* **Fundamental types**
* **Collection types**

**Fundamental type:**

Fundamental types represented class objects it stores the data and python support following fundamental type

* **Int, boot, float, complex, str**

**integers**

Whole numbers from negative infinity to infinity, such as 1, 0, -5, etc.

**float**

Short for "floating point number," any rational number, usually used with decimals such as 2.8 or 3.14159.

**strings**

A set of letters, numbers, or other characters.

**Collection types:**

It represents the class objects it stores the other objects and we called these objects as an element

All collection types represented classes objects are iteral objects

Python support following types of collections

* **List, tuple, set, dictionary**

**Mutable object and immutable object:**

**Immutable objects:**

the objects which does not allow to modify the data are elements of those objects are known as immutable objects

We cannot create two difference immutable objects with the same content.

Applying iterations on the immutable objects takes less time.

All fundamental type repressed class objects take less time and all fundamental type repressed class objects and tuple class object are mutable objects.

**Mutable objects:**

the objects which allow to modify the data are elements of those objects are known as mutable objects

We can create two different mutable objects with in the same content

Applying iteration on the mutable objects takes more time

List, set, dictionary, tuple

**String handling:**

Group of characters are sequence of characters is known string.

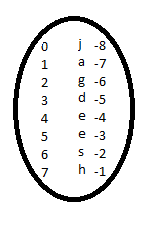
In python group characters are sequence of character we can store in the string class objects.

String class objects can be created by using (‘) single-quoted and (‘’’) triple-quoted.

By using (‘) single quoted we can create string class object with one line.

By using (’’’) triple quoted we can create string class object with one or more lines.

String class objects support both positive and negative index positive index starts from zero (0), negative index starts



**Operators:**

Operators are nothing but constructer which are used to perform the operation on the data of the objects with pointed by the operands are address of the object with are pointed by operators

**Arithmetic operations**

Arithmetic operations are used to perform the mathematical operation like addition, multiplication, division, modules, flow division and exponent operations

(+, -, \*, /, %, \*\*)

Comparison operator are used compare the data of the objects which are pointed by the operands

(<, >, =, !=, <=, >=,)

**Logical operators:(and, or, not)**

Logical operators are used to perform mathematical logical operations, logical AND, logical OR, logical NOT,

**Bitwise operators:**

Bitwise operators convert the data in the form of binary format perform the operation on the binary data and gives the results in the form of decimal format

(&, ^, ~, >>, <<)

**Assignment operators**

Assignment operators are used to assign the data to the operands

|  |  |  |
| --- | --- | --- |
| Operators | Examples | Equivalent to |
| = | X=5 | X=5 |
| += | X+=5 | X=x+5 |
| -= | x-=5 | X=x-5 |
| % |  |  |
| %= |  |  |
| // |  |  |
| &= |  |  |

**Special operators**:

Python supports two types of special operators

1. Member ship operators
2. Identity operators

**Member ship operators**:

Member ship operators are used to search for required element is available or not available is given hittable operator

**Identity operators:**

Identity operators are used to compere the address of the objects which are pointed by the object

**Blocks:**

The set of statement which are following same space with indentation is known as a block

* Block begins when indentation is increases
* Block can contain another block
* Block ends when indentation decreases to zero or to an it’s containing block indentation
* Block should contain at list one statement

**Control flow statements:**

These statements are used to disturb the normal flow of the execution of program

Python supports two types

1. Conditional
2. Looping statement

**Conditional statement:**

Conditional statement is used to decide whether block has to execute are skip the execution block based on the condition

Any expression which returns Boolean values is known as condition

Every condition is an expression but every expression is not acondition**.**

Python supports three conditional statements

**If, else, elif**

**LOOPING statement:**

Looping statements are used to executes set of statement repeatedly python supports two looping statement they are

1. While
2. For

**Collections: list, tuple, set, dict**

All collection types represent the classes objects to store the other objects and we called these objects as an element

All collection types represented classes objects are hiterable objects

Every collection type represent for class provides methods and by using those methods we can perform the operation of the elements of the collections

Python support following types of collection

List, tuple, set, dict

**List:**

list can be created by using “[]” square brackets are by calling list function

list objects are mutable objects the elements of the list can be mutable are immutable objects

duplicate elements allowed in list and hector genres element are allowed in the list.

List supports both positive and negative indexing.

Ex:

x=[]

print(x)

print(type(x))

print(len(x))

y=list()

print(y)

print(type(y))

print(len(y))

z=[10,20,32,33,43]

print(z)

p=[10,20,30,40,50]

print(p)

q=[100,10.225,True,(3+4j),'jagadeesh']

print(q)

output:

RESTART: C:/Users/JAGADEESH/AppData/Local/Programs/Python/Python36-32/loo.py

[]

<class 'list'>

0

[]

<class 'list'>

0

[10, 20, 32, 33, 43]

[10, 20, 30, 40, 50]

[100, 10.225, True, (3+4j), 'jagadeesh']

>>>

Ex:2

x=[10,20,30,40]

print(x)

x.append(50)

print(x)

x.insert(2,69)

print(x)

print(x.count(10))

print(x.index(40))

y=x.copy()

print(y)

z=[10,20,530,140]

x.extend(z)

print(z)

x.remove(40)

print(x)

x.sort()

print(x)

x.sort(reverse=True)

print(x)

x.reverse()

print(x)

output:

RESTART: C:/Users/JAGADEESH/AppData/Local/Programs/Python/Python36-32/loo.py

[10, 20, 30, 40]

[10, 20, 30, 40, 50]

[10, 20, 69, 30, 40, 50]

1

4

[10, 20, 69, 30, 40, 50]

[10, 20, 530, 140]

[10, 20, 69, 30, 50, 10, 20, 530, 140]

[10, 10, 20, 20, 30, 50, 69, 140, 530]

[530, 140, 69, 50, 30, 20, 20, 10, 10]

[10, 10, 20, 20, 30, 50, 69, 140, 530]

>>>

Unpacking elements of list:

x=[10,12.22,True]

print(x)

a,b,c=x

print(a,type(a))

print(b,type(b))

print(c,type(c))

|  |  |
| --- | --- |
| list | Tuple |
| list can be created by using “[]” square brackets are by calling list function  list objects are mutable objects the elements of the list can be mutable are immutable objects | Tuple objects can be created by using parenthesis or by calling tuple function or by assigning multiple values to a single variable |
| Duplicate elements allowed in list and hector genres element are allowed in the list. | Tuple objects are immutable objects |
| List supports both positive and negative indexing. | Tuple supports both positive and negative indexing. |
| List con not be used as a key for the dictionary | Duplicate elements are allowed and hector genres element are allowed in the tuple |
| Every element in the list is represent with unique index. | Tuple can be used as a key for the dictionary if the tuple is storing only immutable elements. |
| In cession order is preserved in list | In cession order is preserved in tuple |
|  | If the frequent operation is retrieval of recommended to use tuple |

Nested list: Nested lists are list objects where the elements in the lists can be lists themselves. A couple of examples will motivate for nested lists and illustrate the basic operations on such lists.

Ex:

x=[(10,20,30),{40,50,60},[70,80,90]]

for p in x:

print(p,type(p))

for q in p:

print(q)

How to search element in the list?

x=[10,20,30,40,50,60]

i=int(input("enter search element:"))

for i in x:

print("element is found")

else:

print("element is not found")

Write a given string in reversing order =‘jagadeesh’

a="jagadeesh"

print(a)

x=list(a)

x.reverse()

print(x)

b=' '

for i in x:

b=b+i

print(b)

o/p: RESTART: C:/Users/JAGADEESH/AppData/Local/Programs/Python/Python36-32/sum.py

jagadeesh

['h', 's', 'e', 'e', 'd', 'a', 'g', 'a', 'j']

hseedagaj

Range: - range is pre-defined function is used to generate values for list comprehension.

List comprehension: - the concept of generating the element into the list object by writing some logic in the list is known as a list comprehension.

X=[p for p in range(10)]

Print(x)

Y=[p\*p for p in range (10)]

Print(y)

Z=[q\*q for q in range (10,20) if q%2==0]

Print(y)

This is a Python Program to find the factorial of a number using recursion.

Ex:1

def factorial(n):

if(n <= 1):

return 1

else:

return(n\*factorial(n-1))

n = int (input("Enter number:"))

print("Factorial:")

print(factorial(n))

what is global variable? And local variable?

Global variable:

The variables which are declared outside of all the functions are as G.v

Global. v of one program can be accessed within all the functions of the same program.

Global. v of a program memory will be allocated only once.

Local variable:

The variable which are declared within the function are know as local variable .

Local variable of one function can’t be accessed outside of that function

For all the local variables of a function memory will be allocated with respect to every time function

Ex:

a=71

def f1():

global a

a=7654

print(a)

def f2 ():

print(a)

f1()

f2()

Anonymous function or lambda function:

A function which doesn’t contain any name is know as a anonymous function of lambda function.

Lambda arguments: expression

Lambda function we can assign to the variable and we can call the lambda function through the variable.

Ex:

def cube(a,square):

sq=square(a)

return a\*sq

x=cube(10,lambda b:b\*b)

print(x)

filter function:- filter function gives the values ,based on the condition whether it is true or false.it displays only true values

map function:- map gives the value whether it is true or false.

It display both true and false.

**Interview points:**

**Merging two sorted list.?**

**Ex :1**

**a=[1,2,3,4,6]**

**b=[9,7,6,5,11,33]**

**c=[]**

**while a and b:**

**if a[0]<b[0]:**

**c.append(a.pop(0))**

**else:**

**c.append (b.pop(0))**

**print(c+a+b)**

**ex:2**

**a=[1,2,3,4,6]**

**b=[9,7,6,5,11,33]**

**a.extend(b)**

**c=sorted(a)**

**print(a)**

**output:**

**[1, 2, 3, 4, 6, 9, 7, 6, 5, 11, 33]**

**>>>**

**[1, 2, 3, 4, 6, 9, 7, 6, 5, 11, 33]**

**>>>**

**Initializing dictionary with list -1**

**In the following example, we make a dictionary like {‘citys’:[states],…],}**

**cities={'hyd':'tg', 'chi':'tn', 'bng':'ka' ,'mub':'mh'}**

**from collections import defaultdict**

**d1= defaultdict(set)**

**for k,v in cities.items():**

**d1[v].append(k)**

**print (d1)**

**d2={}**

**for k,v in cities.items():**

**d2.setdefault(v,[]).append(k)**

**print(d2)**

**output:**

**>>>**

**defaultdict(<type 'list'>, {'tn': ['chi'], 'tg': ['hyd'], 'ka': ['bng'], 'mh': ['mub']})**

**{'tn': ['chi'], 'tg': ['hyd'], 'ka': ['bng'], 'mh': ['mub']}**

**>>>**

**Now we want to make a dictionary with the number of digits as the key and list of numbers the value:**