**Python IDEs:**

**Pycharm –Best IDE**

Spyder

Eclipse

**Advantage of Python:**

1. Concise code
2. Rich Libraries
3. Easiness to programmer
4. Freeware

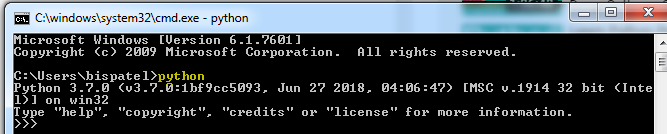
Python is an interpreted language. Python VM has inbuilt functionality to compile.

**How to install Python:**

1. <https://www.python.org/>



1. Python 2 and Python 3 are different and not compatible. Python 3 is developed as independent language. Highly recommended to go for Python 3. After 2020 there won’t be any support for Python 2.
2. To check if python is installed:



1. Environment variable:

C:\Users\bispatel\AppData\Local\Programs\Python\Python37-32; C:\Users\bispatel\AppData\Local\Programs\Python\Python37-32\Scripts

1. **Python REPL (Read Evaluate Print Loop) Tool**: Python IDLE

Without writing big code we can check the logic

1. To open python in command line: **py**

Ex: Read a number from command prompt and write Hello that many times.

1. Create a .py file
2. To run a .py file Run🡪Run Module or click “F5”.
3. To run from command prompt:

Python filename.py

Py filename.py

**To read a number:**

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

**To loop through:**

for i in range(n):

print("Hello");

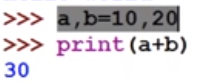
**To print a line:**

Print(“Hello”);

**Python Module:**

1. Core Python
2. Advanced Python

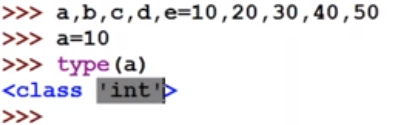
To add 2 numbers:



10 and 20 will be assigned to a and b respectively.

C, Java === Statically Typed Programming Language

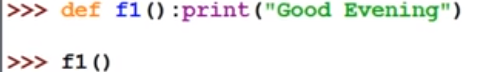
Python== Dynamically Typed Programming Language



Types will be assigned automatically based on the provided value.

Syntax from C and ABC language.

Functional Programming:



**Python** is a general purpose high level programming language (Human readability)

Guido Van Rossam =🡺1989 developed. 1991 it was available to public.

**Where we can use Python:**

1. Desktop Application
2. Web Application

Django is the framework to develop web application in Python.

1. Database Applications
2. For Networking application
3. Games
4. Data Analysis
5. Machine Learning
6. AI
7. For IOT applications

**Python features:**

1. Functional Programming from C
2. OOP from C++
3. Scripting language features from Perl and Shell script
4. Modular programming features from Module-3

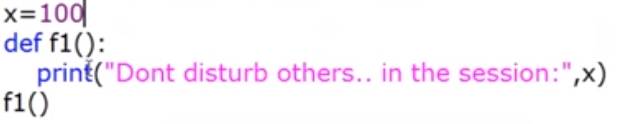
**Advantages**:

1. Simple and easy to learn
2. In Java 53 keywords are there. However, In Python only 30 keywords.
3. Free ware (We are not required to pay anything for license) and open source (We are able to see the source code)
4. High level programming language.
5. Platform Independent. (Write Once and Run Anywhere)
6. Portability: Moving from one platform to another without changing anything.
7. Dynamically typed program (Based on the value provided to a variable, the type will change dynamically)
8. Both Procedure Oriented and Object Oriented.
9. Interpreted language
10. Extensible ---Native method support is there. Whatever other languages code we can use in Python. In Java we have some code with 1 lakh line. We can use python to improve the performance.
11. Embedded
12. Extended libraries

**Limitations**:

1. Performance wise not good as it is interpreted language.
2. Mobile applications are not there.

Without having the class, we can call the function



**Types of Python:**

1. **CPython:** Can work with C language
2. **JPython:** Python for Java application
3. **IronPython**
4. **PyPy**
5. **RubyPython**
6. **AnacondaPython**
7. **Stackless (Python for concurrency)**

**Python Versions:**

Python 1.0 introduced in Jan 1994

Python 2.0 introduced in October 2000

Python 3.0 introduced in December 2008 (Recommended to use)

Python 3.0 does not provide backward compatibility

Python will be supported till 2020

**Python 3 is very popular and it is growing like anything**

**Identifiers:**

Name in Python program. (Variable, Method, Class)

Rules to define identifiers in Python:

------------------------------------------------

1. Alphabet symbols (both upper and lower case)
2. Digits (0 to 9)
3. Underscore (\_)
4. Identifier should not start with digit
5. Identifiers are case sensitive
6. Keywords cannot be used for variable purpose.
7. Max length allowed for python identifier: No limit

Identifier starting with “\_” it indicates Private

Starting with 2 underscores “\_\_” it indicates strongly private

Start and end with 2 underscores: Language specific variable defined by python.

**Reserved words:**

33 reserved words

True, False, None,

and, or, not, is,

if, else, elif,

while, for, break, continue, return, in, yield,

try, except, finally, raise, assert,

import, from, as, class, def, pass, global, nonlocal, lambda, del, with

**continue:** Skip current execution

**raise** is equivalent to throw in java

**Note:**

* Only Alphabet symbols
* Except first 3 all remaining contains lower case alphabet symbol.

**To know all the keyword**

Import keyword

keyword.kwlist

**Data Types:**

1. int
2. float
3. complex
4. bool
5. str
6. bytes
7. bytearray
8. range
9. list
10. tuple
11. set
12. frozenset
13. dict
14. None

**Python provides some inbuilt function:**

1. print ()
2. type ()
3. a=10, id(a)🡪 to print object reference

**In Python everything is an object.**

1. **int ():** Integral values 10,20,30,1000000000
2. Decimal form (Base-10,0-9) Ex: 7878
3. Binary form (Base-2,0-1) Ex: 0b1111,0B1111==15
4. Octal form(Base-8,0-7) Ex:0o111==73
5. Hexadecimal (Base-16, 0-9 and a-f or A-F) Ex: 0x

Size of object is not fixed in any language. In Python everything is an Object. So, size/range/maxvalue/minvalue concept is not there in python.

Base Conversions:

bin () ---To convert the value from any base to binary

oct ()—To convert the value from any base octal

hex () =To convert the value from any base to Hex

**Python data type was there is Python 2 but not in Python 3**

1. **Float ()**

For floating type datatype, we can use only decimal value

1. **Complex ():**

a+bj===a is real part and b is imaginary part.

J^2=-1

Real part can be anything but the imaginary part can only be decimal value.

X=10+20J

X.real =10

X.imag =20

1. **bool** === True /False

True is equal to 1

False is equal to 0

True+True ==2

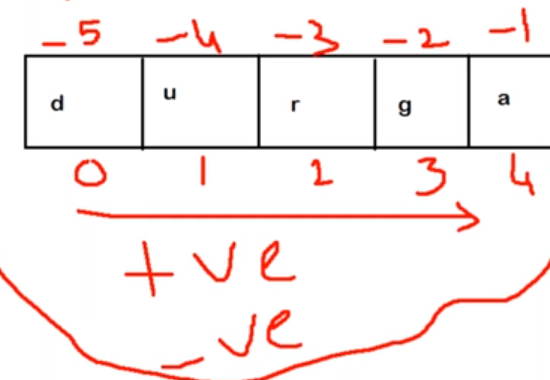
True+False ==1

True/False == ZeroDivisionError

1. **str**

Recommended to use single quote

Triple Single quote or double quote are used to represent string in multiple line



**Slice:**

S[begin:end]

Begin and end are optional

S[begin:end:step]

**Repetition Operator:**

S=’Bishwa’

S\*3 =’BishwaBishwaBishwa’

**Length**:

Len(s)-🡪6

**Python’s fundamental data types:**

int, float, complex, bool, str

There is no Unicode concept in Python as char datatype is not there

**Type Casting or Type coersion:**

int ()

float ()

complex ()

bool ()

str ()

1. **int ():**

int (123.456) ===123

int (True) ==1

int (“10”) ==10

int(“10.6”)== ValueError

We can’t convert complex number to int

ii. **float ()**

float (10) ==10.0

float(10+20j) ==>TypeError

float(True) ==1.0

float(False) ==0.0

float ("10.5") ==10.5

float(“ten”) === ValueError

iii. **complex ()**

1. complex (a) ===a+0j
2. complex (a, b) === a+ bj
3. complex (True) == 1+0j
4. complex (False) ===0j
5. complex ("10") === 10+0j
6. complex ("10.5") ====10.5+0j
7. complex (10,20) ==10+20j
8. complex ("10","20") == TypeError. Complex can’t take second arg if first is a String

iv. **bool ()**

0 means False and others are true

bool (0) == False

bool (10) ==True

bool (-10) == True

bool (10.6) ===True

bool (0+0j) ==False

If both real and imaginary part are 0 then False else it is True

Bool (‘’) == True

If empty then False else it is True

v. **str ():**

**Immutable vs Fundamental Data Type:**

Mutable == Changeable

Immutable == Non- Changeable

**All fundamental data types are immutable**

To print the address of a variable

x=10

Id(x)

X=10

Y=10

X **is** Y === True

Is operator is used for reference check

X=256

Y=256

X is Y===True

X=257

Y=257

X is Y===False

For **int** reusing the same object is possible from **0 to 256** range. 0-256 because this is range for common use of int.

For **floating** type data type reusing concept is not there.

X=10.0

Y=10.0

X is Y ===False

For **Complex** type data type reusing concept is not there.

**int** === 0 to 256(Most commonly used range). At the time python interpreter starts: from 0 to 256

integer objects will be created at the beginning.

**bool** === always

**str** === always== **String literal is the most commonly reused concept in programming**

**float === no reusing concept**

**complex === no reusing concept**

**immutable** == once we create an object. We can’t change that.

vi. **bytes**

1. bytes must be in the range of 0 to 256.
2. immutable
3. Used to represent binary data

vii. **bytes array**

both are same. But it is mutable

viii. **list**

List representation l= [a, b, c]

Group of values as single entity. Insertion order is preserved and duplicates are allowed.

Heterogeneous objects are allowed. Growable in nature

Index 🡺 +ve or -ve

+ve ===Left to Right

-ve === Right to Left

\* == Repetition operator

S= [10]

S1= S\*S = [10,10]

ix. **tuple**

List and tuple are same except list is mutable but tuple is immutable

**Tuple representation:**

t = (a, b, c)

x. **range**

range data type represents a sequence of values. Elements in range are **immutable**.

**Form1**: range(end)

It represents values from 0 to end-1

for I in range(10):print(i)

0 1 2 3 4 5 6 7 8 9

**Form2**: range (start, end)

To represent number from start to end-1

**Form3**: range (start, end, step) : Not applicable for float

Ex: range (10, 50,5)

10,15,20,25,30,35,40,45

**xi. set**

No duplicates

No ordering

Representation: {}

No indexing concept, no slicing as order is not there

It is mutable

Creating an empty set : set()

List ===append is used

Set === add is used

xii. **frozenset**

Like set except it is immutable

**xiii. dict**

Group of key value pair:

>>> d={100:'A',200:'B'}

>>> type(d)

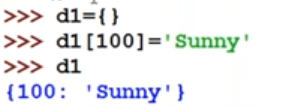
<class 'dict'>

**Representation: {}**

By default, it is treated as dictionary. It is Mutable.

To add elements to dict:

d[key] = value



**Quick Review:**

1. Fundamental data-types: int, float, bool, str, complex
2. bytes and bytearray 🡺 to represent binary data like images, video files and audio files
3. long data type

Python 2🡪 Available

Python 3🡪 Not available

1. In Python there is no character data type
2. int: To represent whole numbers integral numbers. Immutable
3. float: immutable
4. complex: immutable
5. bool: True-1, False-0 immutable
6. str: immutable
7. bytes: 0 to 256 range

l = [10,20,30] b=bytes(l)

immutable

1. bytearray: mutable
2. range: To represent range of values. Immutable
3. list: Maintain insertion order, Allows Duplicate, Heterogeneous objects, growable, [], mutable
4. tuple: same with list. Immutable
5. set: order not maintained. Mutable. Index concept is not available.
6. frozenset: Immutable
7. s = {}🡪 Internally it is treated as dict
8. dict🡪Mutable
9. None data type:

If a function does not return anything.

Ex: def f1(): a=10

Print(f1())🡪None

**Escape characters:**

**\n, \t, \r, \b (backspace), \f (form feed), \’, \”, \\, \v**

**Constant: No constant concept**

**Operators:**

1. **Arithmetic Operators**

+ ==> Addition (Applicable for String also). If we want to apply “+” operator in String then both argument must be String.

Ex: “Hello”+3-🡪Error

- ==>Subtraction

\* ==>Multiplication

/ ==>Division Operator (Always work for floating point arithmetic)

% ==>Modulo Operator

// ==>Floor Division Operator (Result can be either int or float depending on the argument type)

10.5//2===5.0

10//2==5

\*\* ==> Exponent Operator or Power Operator. 2\*\*3=8

10\*\*-2==0.01

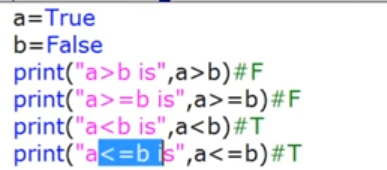
ZeroDivisionError == Anything divided by 0 will give this error

1. **Relational Operator or comparison operator**

A==65

a==97

We can use relational operator for String, Boolean



10<20<30: Chaining of relational operator. If at least one comparison fails the result is always false

Equality will be checked with same data type.

== is used to check the content.

“is” used for reference check.

= assignment

== comparison operator

1==True🡪True (boolean can be converted to int)

1. **Logical Operator**

and/or/not

0 means False

Non-zero means True

Empty String🡺False

Rule: x and y=== If x evaluates to false then result is x otherwise returns y

Ex: 10 and 20==10 is non-zero so True, according to rule x is true. So, output will be y i.e. 20

1 and ‘abc’--🡪 ’abc’

0 and ‘test’🡪 0

10 or 20🡪10

0 or 20🡪20

not 10🡪False

not ‘’🡪True

not 0🡪True

1. **Bitwise Operator**

& 🡺If both bits are 1 then only 1 otherwise 0

| 🡺 If at least one bit is 1 then 1 otherwise 0

^ 🡺 x-or (If both bits are different 1 otherwise 0)

~ 🡺bitwise complement operator 1🡺0, 0🡺1

<< 🡺bitwise left shift

>> 🡺bitwise right shift

applicable only for int and Boolean

Ex: 4&5== 100 & 101==100==4

4|5==100 | 101==101==5

4^5==100 ^101==001==1

~4==~00000…….100==111111…..011==2s’ complement===000…..100+1=000…101=-5

Shift Operator:

<< Left Shift Operator:

10<<2🡺40

>> Right Shift Operator

10>>2🡪2

Left hand side vacant cells sign bit

+ve number =🡺0

-ve number 🡺1

1. **Assignment Operator**

++x is treated as sign only not increment operator, In Python there is no assignment operator.

+=, -=, \*=, /=, %=, //=, \*\*=, &=, |=, ^=, >>=, <<=,

a&=5 is equivalent to a=a&5

**Ternary Operator**:

x= first value if condition else second value

Ex: x=30 if 10<20 else 40

print(x)

**Swapping Operation**:

a, b = b, a

To read input from keyboard:

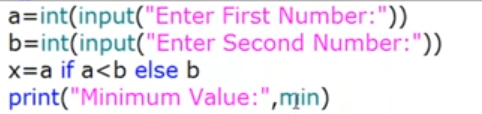
input () ===The data read is considered as String

Converting string to int type:

Int (str)

Ex: int (input (“Enter a number”))

Ex2:



Nesting of conditional operator:

X=1st Value if condition else 2nd Value if condition else 3rd Value

Ex:

X=10 if 20<30 else 40 if 50<60 else 70

1. **Special Operator**
2. Identity operators:

If r1 and r2 pointing to same object then r1 is r2 will return true

“is” 🡺Address comparison

“==” 🡺Content comparison

1. Membership operators:

list1 = [10,20,30]

print (10 in list1)🡪 True

print (50 not in list1)🡪 True

**in, not in**

**Modules:**

A group of functions, variable, classes.

Library: Group of Modules

Math Module

Ex: import math as m

print(m.sqrt(25))

print(m.pi)

To use the member name directly: Not recommended to use

From math import \*

Print(sqrt(16))

Print(pi)

**Functions**:

Sqrt (), ceil (), floor(), pow(x,y), factorial(), gcd (), sin(), cos()

p=3.14

e=2.71

Inf=infinity

nan = not a number  
**Input and output Statement:**

**Read dynamic data from the keyboard:**

**Python-2**

1. **x= raw\_input(“Enter the input:”)🡺str**

we required to use typecasting functions

1. **y= input (“Enter some number”)**

Not required as str type whatever type provided. We are not required typecasting.

**Python-3**

We should use only input function

**How to read input?**

eno = int (input("Enter Employee Number:"))

ename = input ("Enter Employee Name:")

esal = float (input ("Enter Employee Salary:"))

eaddr = input ("Enter Employee Address:")

married = bool (input ("Enter Employee Marriage Status: [True|False]"))

print ("Please confirm Information...")

print ("Employee Number:", eno)

print ("Employee Name:", ename)

print ("Employee Salary:", esal)

print ("Employee Address:", eaddr)

print ("Employee Marriage?:", married)

**How to read Multiple values from keyboard in a single line:**

a, b=[int(x) for x in input ("Enter 2 numbers:"). split (" ")]

print(a)

print(b)

**Q. Read 2 float values from keyboard which are specified with, separation and print sum**

x,y=[float(x) for x in input("Enter 2 float numbers:").split(",")]

print ("Sum:", x+y)

**eval function is used for evaluation**

eval(10+20-15)🡪 5

eval automatically detect the type

**Ex1: Loop through the data**

n = input ("Enter some data:")

x=eval(n)

print (type(x))

for x1 in x:

print(x1)

**Ex2: Different datatype**

a, b, c =[eval(x) for x in input ("Enter 3 values:").split(",")]

print(type(a))

print(type(b))

print(type(c))

**Command Line Arguments:**

We can access command line argument using “argv”. It is available in “sys” module

from sys import argv

print(type(argv))

Name of the python file is the first command line argument

Ex:

test.py

from sys import argv

print(type(argv))

print(argv)

>>py test.py 10 20 30

[‘test.py’, ‘10’, ‘20’, ‘30’]

**Output statement:**

1. print ()—Without any argument
2. print (string)

+🡺 Both argument should be str type only

\*🡺one argument should be str and other should be int

1. print (“a”+”b”) 🡪ab (no space)

Print(“a”,”b”)🡪 a b (space)

1. a,b,c =10,20,30

print(“The values are:”,a,b,c)

o/p: The values are: a b c

1. sep attribute

print (a,b,c,sep=”:”)🡪a:b:c

1. By-default print is new line

To print in one line

print(“hello”,end=” “)

print(“world”,end=” “)

o/p: hello world

end attribute is used in between multiple line.

1. %i === int type

%d===int type

%f === float type

%s=== String type

print (“formatted string” %(variable\_list))

Ex1:

a, b, c=10,20,30

print ("a value is %i and b value is %i" %(a,b))

Ex2:

>>> name="Bishwa"

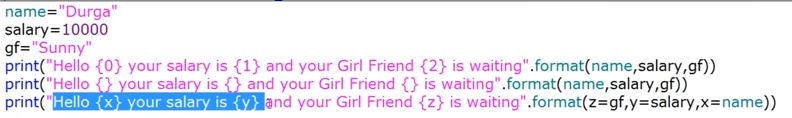
>>> l= [10,20,30,40]

>>> print ("Hello %s the list is: %s" %(name,l))

Hello Bishwa the list is: [10, 20, 30, 40]

**Replacement Operator:**

{} 🡪 Replacement operator



**Flow Control:**

1. **Conditional/ Selection statement:**

If

If-else

If-elif-else

If-elif

1. **Iterative statement**

loops

for loop==based on condition we can use break, continue, pass

while

1. **Transfer Statements:**

Break, continue, pass

Ex: for eachElement in sequence:

Perform action

**Note\*:** Increment and Decrement operator not present in Python.

**If we know iteration in advance: for loop else we should go for while loop**

**break:**

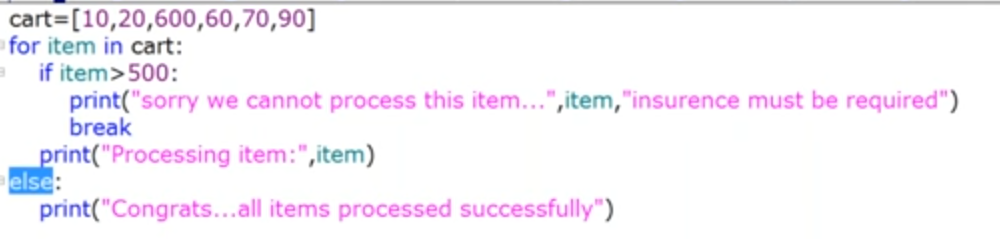
Based on some condition if we want to break loop execution.

**Else in Loop:**

**For-Else**

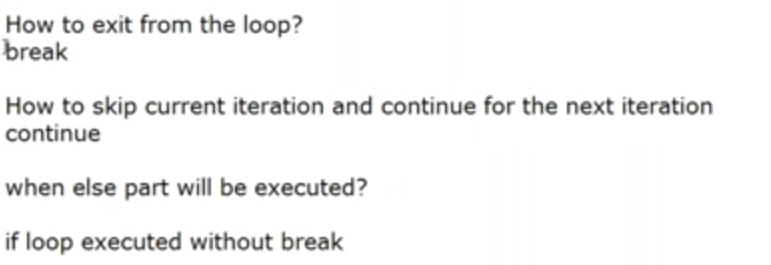
If the loop executes successful then else statement will be executed. If break is executed then else block won’t be executed.

Else means loop without break.

**Ex: Else part won’t be executed here**

**Ex2: Else block will be executed**





**Pass Statement:**

pass is a keyword in python. Where ever empty block is present we should go for pass.

Empty statement won’t do anything

**del statement:**

del variable

**String Data Type:**

There is no character data type in python.

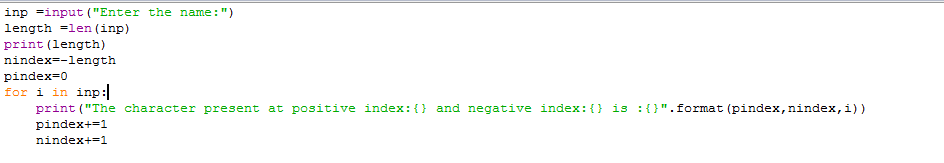
literals –String values

**multi line string literals:**

triple quotes

Escape character: ”\”

**Read a String and print the character along with +ve and -ve index:**



Access character from String:

1. By using index
2. By using slice operator

+ve index: left to Right

-ve index: Right to Left

**Slice Operator:**

**s[begin:end:step]**

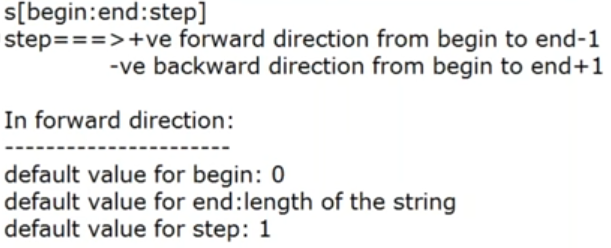
If step operator is +ve then left to right, -ve right to left

S=’0123456789’

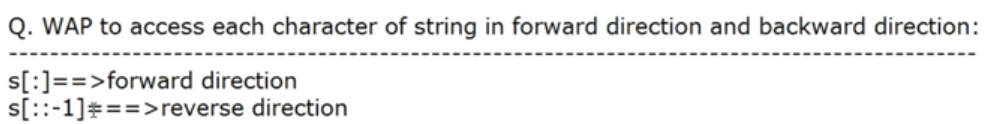
S[2:8:1] 🡺234567

S[2:8:-1]=’’

S[8:2:-1]🡺876543



**Mathematical Operator:**



**a 🡺 97**

**A🡺 65**

In general, == is used for content comparison and “is” operator is used for reference comparison.

**To remove the space from string:**

lstrip ()

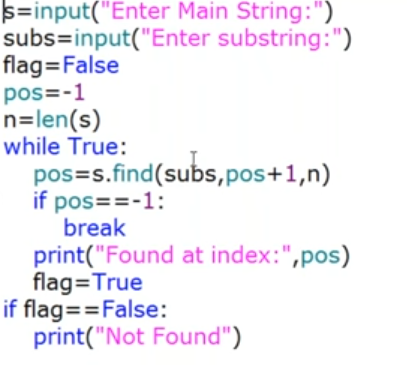
rstrip ()

strip ()

Ex:



**To find a Sub String in a String:**



**Counting Sub String in the given String:**

1. str.count(substring)
2. str.count(substring,begin,end)🡺 From begin index to end-1, how many times substring came

**Replacing a String:**

1. str = str.replace (“original String”, ”Replaced One”)

**Splitting of a String:**

1. str =str.split(“splitBy separator”)
2. str =str.split(“splitBy separator”,max occurance)

Default split by is: Space

**Differentiate between split and sep:**

Split is for String

Sep is used while printing

Ex:

s ="10,20,30,40,50,60,70,80,90"

l=s.split(",")

l == ['10', '20', '30', '40', '50', '60', '70', '80', '90']

l=s.split(",",3)

l== ['10', '20', '30', '40,50,60,70,80,90']

l=s.rsplit(",",3)

l== ['10,20,30,40,50,60', '70', '80', '90']

**Join Method:**

T= (“10,20,30”)

S= “:”.join(T)

Print(s)🡪 10:20:30

**Changing Case of String:**

Upper ()—To convert to upper case

Lower () – To convert to lower case

Swapcase

Title – If the first letter of all substring is capital.

Capitalize –Only first letter should be capital

**Checking starting and ending part of string**

s.stratswith

s.endswifth

**How to check alphabets:**

**s.isalpha()**

Check for digit: **isDigit()**

**sorting a string:**

for x in sorted(s1):

out = out+x

**To find the Unicode value from character:**

ord (char) 🡺Unicode

chr (unicode) 🡪Character

**Number of ways to create list:**

1. l= []
2. l= [10,20,30,40]
3. l= eval (input (“Enter some list:”))
4. l= list (sequence)
5. l = s.split (separator)

Important functions of list:

Methods

Functions

Function declared inside a class is called Method

Length of list: len (list)

To get the index of first occurrence: list.index(value)

Count () to get the number of occurrence

**Manipulating elements of list:**

l.append(element)

l.insert(index, element)

**append () and insert ()**

append method is used to add element at last. Insert method is used to add at a specified location.

**Adding a list to another list**

List1.extend(list2)

**To sort elements in a list:**

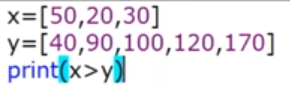
List.sort()

**To reverse the list:** list.reverse()

Comparing List of Objects:

x==y

1. Number of elements must be equal
2. Order should be same
3. Content should be same (including case)



First element of both list will be compared.

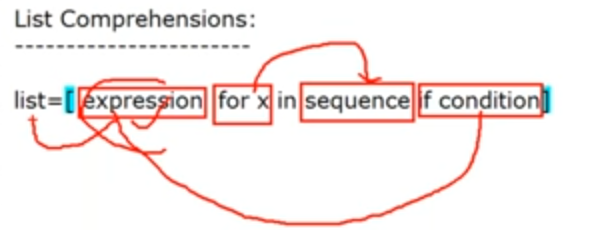
Remove function:

1. remove ()
2. pop ()
3. pop (index)
4. clear () --- To remove all values

**List Comprehension:**

list = [x\*x for x in range (1,11)]

print(l1)=🡺1,4,9,16,25,36,49,64,81,100



**Tuple:**

T= (10) 🡪int type

T = (10,)🡪 tuple

**Important functions of tuple:**

1. Length: len (tuple)
2. count(x) ==tuple.count(element)
3. index () == tuple.index(10)
4. sorted (): To sort elements based on natural sorting order.

**Ex:** t = (30,50,20,10)

Sorted(t)

**Sorting the tuple:**

tuple (sorted(tupleName,reverse=True))

To find the min and max value of tuple:

Min(tuple)

Max(tuple)

**Tuple packing and unpacking:**

**Packing:**

A=10

B=20

C=30

T= A, B, C

**Unpacking:**

T = (10,20,30)

A, B, C = T

Tuple comprehension is not supported in python

List====Mutable

Tuple ==== Immutable

Set ====

a = {}

print(type(a))

a = set ()

print(type(a)) ===== set

s = {10,20,30,40}

l = [50,60,70]

s.update(l)

Adding a single element: add

Adding a group of elements: update

**copy ()**

**add ()**

**update ()**

**pop () ==** To remove and return the element some random element

**remove (x)**

**discard(x)===** If x is not available we won’t get any error

**clear () ===** To remove all elements from the set

**Mathematical Operations:**

1. union ()

Ex: s1. union (s2) or s1 | s2

1. s1. intersection (s2) or s1&s2
2. s1. differences(s2) or s1-s2
3. symmetric\_difference : excluding common objects

**Dictionary Data Structure:**

List, Tuple, set == Only to hold individual objects

Key-value pairs

Dictionary

How to create dictionary:

D= {}

D = dict ()

List, Tuple, set used to handle individual objects.

**How to delete elements from the dictionary?**

del dict

**How to specify multiple values for the single key:**

list=['A','B','C']

>>> d= {100: list}

>>> d

{100: ['A', 'B', 'C']}

1. dict ()
2. d = dict ({})
3. d.get(key)===If not available “None”
4. d.get (key, defaultvalue)
5. d.pop(element)
6. d.popitem() === Will remove some key value randomly

To get all keys:

d.keys () === Type is dict\_keys

d.values ()

items ()===list [(k1,v1),(k2,v2),(k3,v3)]

for k,v in d.items():

print(k,”…………..”,v)

copy ()=== To create exact dictionary copy

setdefault(k,v)=== If the key is not available then key will be added

For huge amount of information: Use Files

Comprehension:

S = {x: x\*x for x in range (1,6))

Print(S)

**Functions:**

def wish (name):

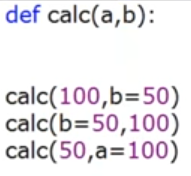
print (“Good Morning:”, name)

wish(“Bishwa”)



Functions defined inside the class is Method. Function can return anything.

If function does not return anything and we print that function. None will be printed.



**Parameters order:**

Positional argument followed

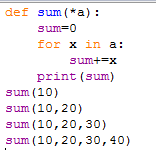
Keyword

Default

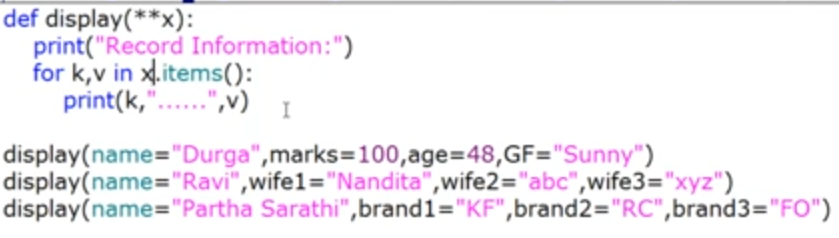
Var

**Variable length argument:**

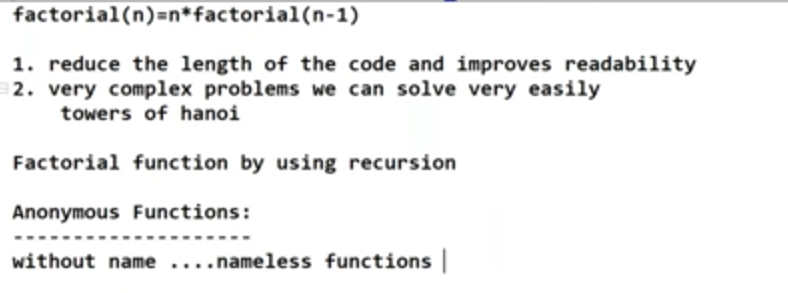
\*a



**Var arg for Strings:**



Anonymous function:



Normal function:

def sqaureIt (n):

return n\*n

s = lambda n: n\*n

**lambda function:**

lambda input: expression

Ex: s = lambda n: n\*n

print (s(4))

print (s(5))

filter ()

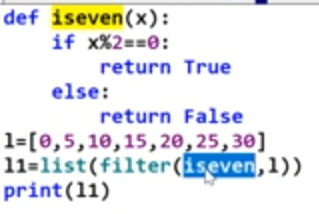
map ()

reduce ()

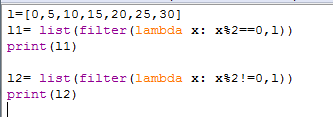
1. **filter ():**

Syntax:

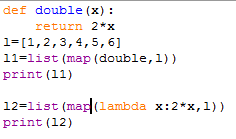
filter (function, list)

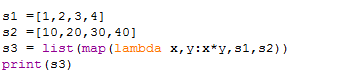


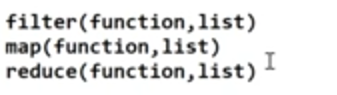
**With Lambda:**



**Map Example:**

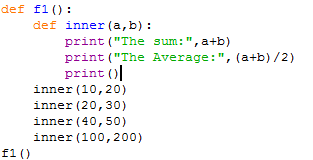




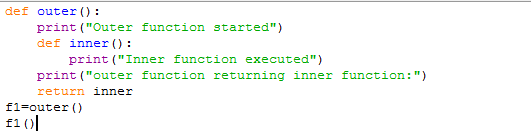


**Nested Function:**

To define function specific repeatedly used function



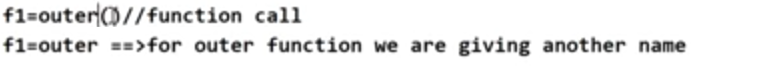
**A function can return another function**



**Output:**

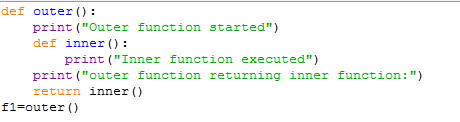


**Function Aliasing:**

****

**With Parenthesis: function call**

**Without parenthesis : Aliasing**

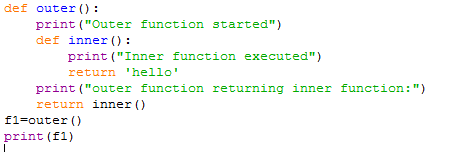


**o/p:**



**Return inner () will call the inner function and the return type will be of inner function**

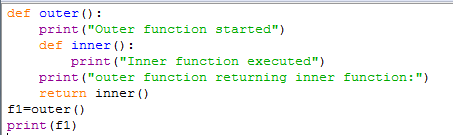
**Ex1:**



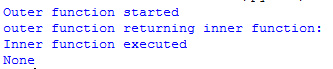
**o/p:**

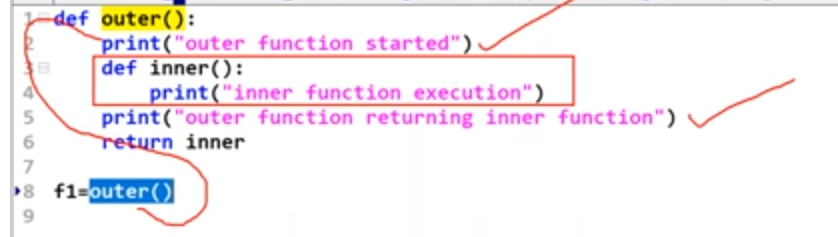


**Ex2:**

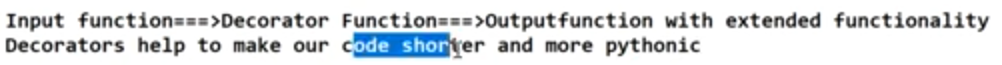


**o/p:**

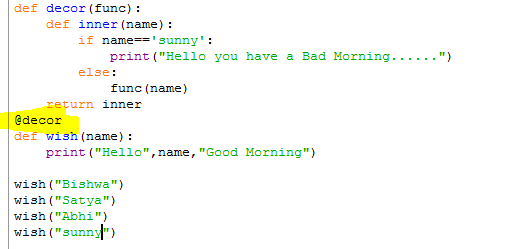




**Function Decorators:**

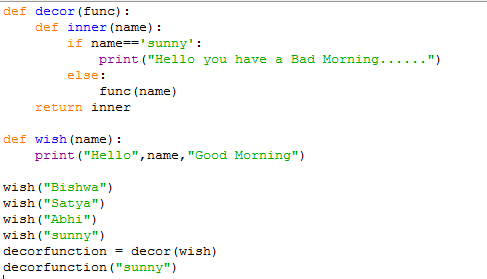


**Ex:**

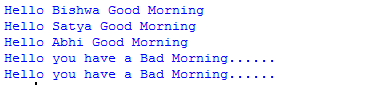




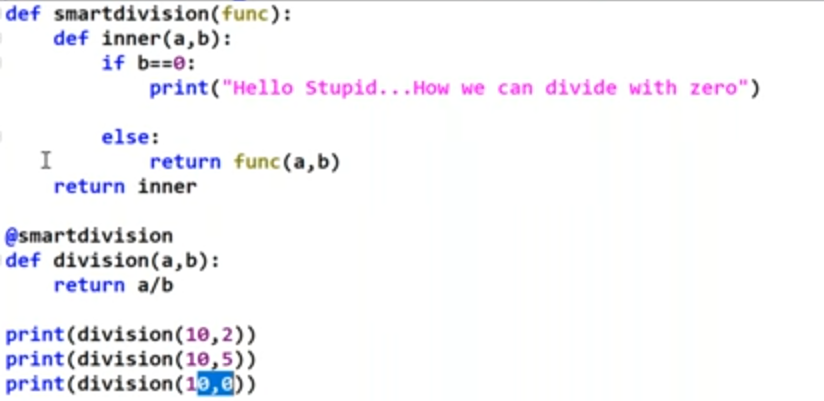
**Ex2:**



**o/p:**



**Ex3:**



**Module:**

A group of functions and variables saved to a file.

Code Reusability

Precise code

Readability

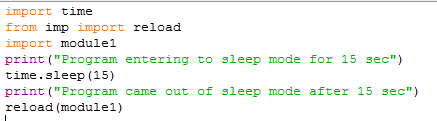
Maintainability

**Syntax:**

from moduleName import method1, method2, method3

PYC file: Python compiled file will be stored in \_pycache\_

**Reloading a module**



**reload function is available in imp module**

**dir()==**List down all the current module members

A module can be executed either directly or from another module

\_\_name\_\_ will decide which one is running

Working with math module:

Defines several functions for math operations:

sqrt ()

ceil ()

floor ()

sin ()

tan ()

pow ()



import math

help(math)

**Working with Random Module:**

1. **random** (): generate some float values between 0 and 1. Not inclusive

from random import \*

for I in range (10):

print(random())

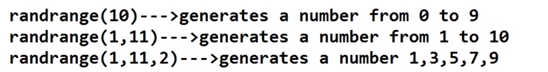
1. **randint**():

randInt(1,100)

1. **uniform** (): Between 1 to 100 floating values

randint(1,100)====int ====inclusive

uniform(x,y)==== float ==not inclusive

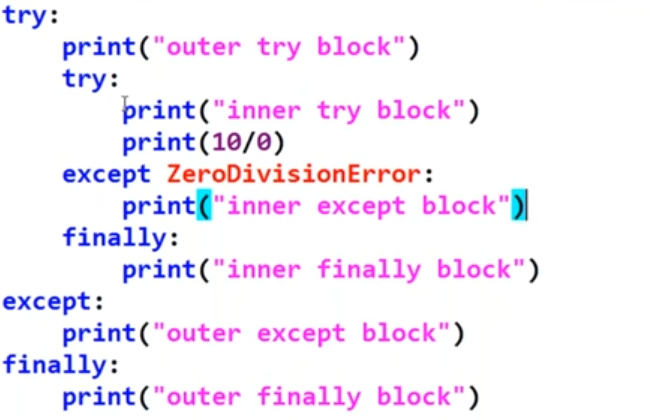


**Try**

**Except**

**Finally**

**Control flow in try-except-finally**



o/p:

outer try block

inner try block

inner except block

inner finally block

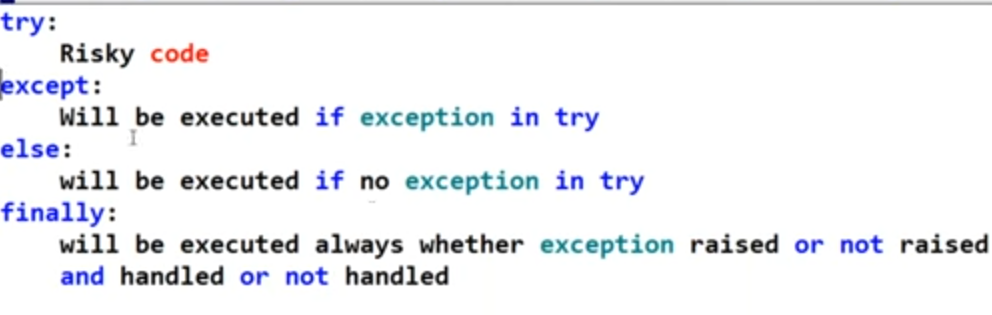
outer finally block

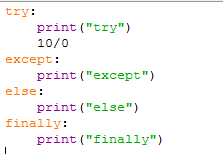
We can use else block with try-except-finally. This block will be executed if there is no exception

Difference between else and finally:

Else will be executed if there is no exception.

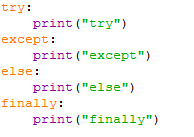
“finally” will always be executed.





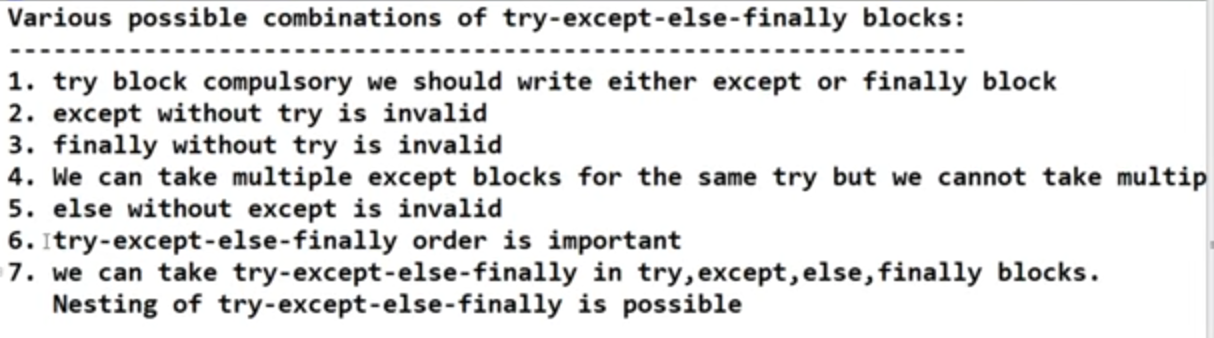
o/p:

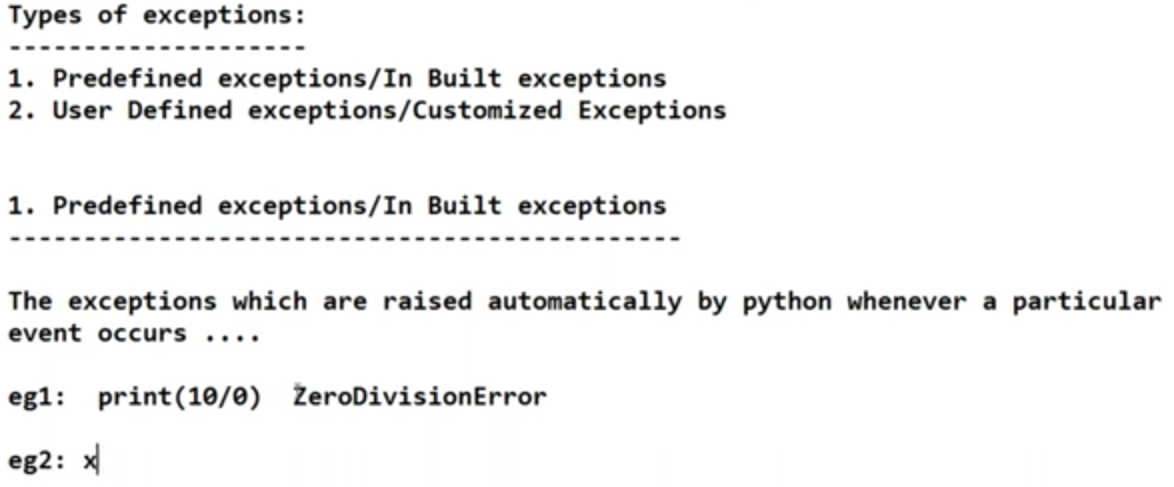




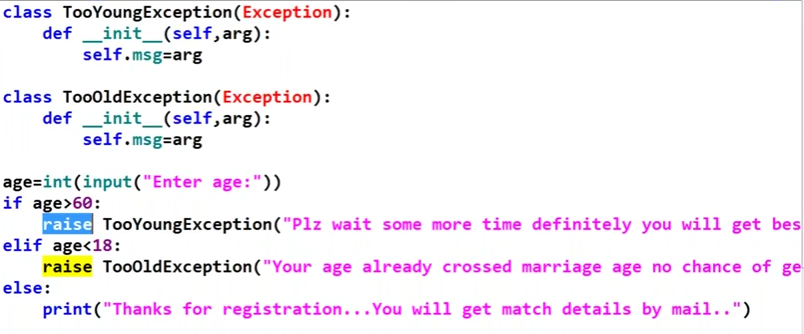
o/p:







**Customized Exception:**



**Logging:**

Inbuilt logging functionality is available.

**Logging level:**

**6 levels**

1. **CRITICAL 🡺 50**
2. **ERROR 🡺 40**
3. **WARNING 🡺 30 (Default logging level)**
4. **INFO 🡺 20**
5. **DEBUG 🡺 10**
6. **NOTSET 🡺 0**

By default, first 3 are available.

So by default warning or higher level will be displayed.

**How to implement logging:**

Name of the file

Level messages

basicConfig () of logging module

logging.basicConfig (filename = ‘log.txt’, level =logging.WARNING)

logging.debug(message)

logging.info(message)

logging.warning(message)

logging.error(message)

logging.critical(message)

**Debugging:**

Defect/bug: Mismatch between expected result and original result

assert statement helps in debugging. These won’t be executed in client machine

**Pickling:**

Process of saving the state of an object to a file is called pickling.

Process of reading state of an object from file: unpickling

pickle module

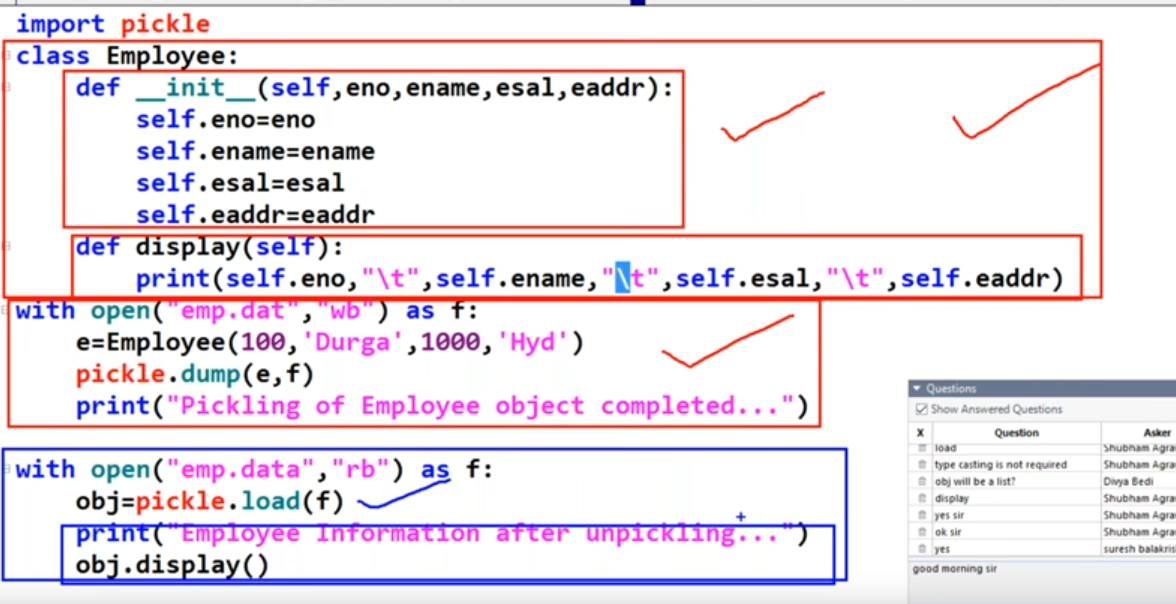
module: pickle

**pickling**:

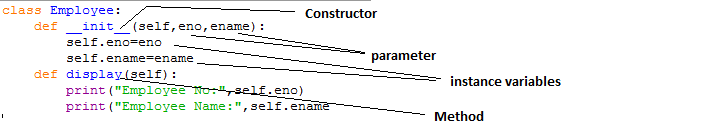
pickle.dump (object,file)

**unpickling**:

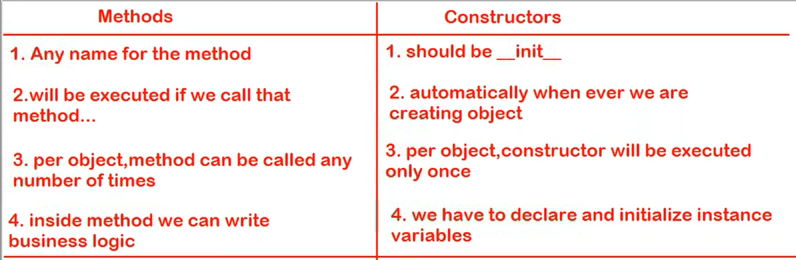
pickle.load (file)



**Creating a class:**



Self always points to current object. It is implicitly used by python



**Types of variables:**

3 types of variables

1. Instance variable (non-static)
2. Static/ class level variables
3. Local variable

**Instance Variable (Non- static)**

1. If the value of variable is varied from object to object
2. For every object a separate copy of instance variable will be created.
3. In general, inside constructor we must declare by using self
4. By using object reference if we perform any changes to instance variable, changes won’t be affected to remaining objects.

**How to access instance variable**

1. Within class using self
2. Outside class by using reference variables

How to know number of instance variable of an object:

objectRef.\_\_dict\_\_

**To delete an instance variable**

del objectRef.variable

**Static Variable:**

Class level

Only one copy will be created

**How to access static variable:**

Either using object reference or class Name

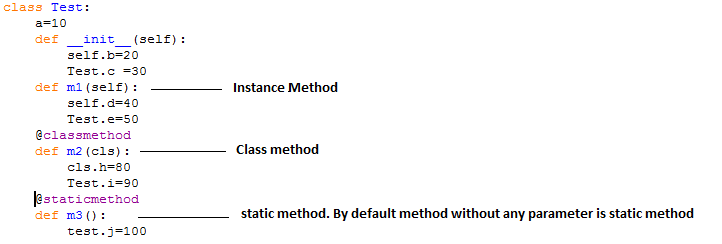
**Places to declare static variable:**

1. Within class directly
2. Inside constructor by using class name
3. Inside instance method by using class name
4. Inside classmethod using cls variable or class name
5. Inside static method by using class name
6. From outside of class by using classname

**@classmethod : To make a method as class method**

To know instance variables: objectRef.\_\_dict\_\_

To know the static variables: ClassName.\_\_dict\_\_



**Regular Expression:**

To represent a group of strings according to a pattern then we should go for Regular Expression.

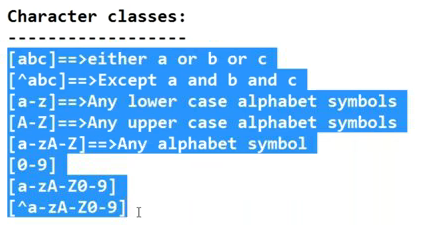
1. compile () function:

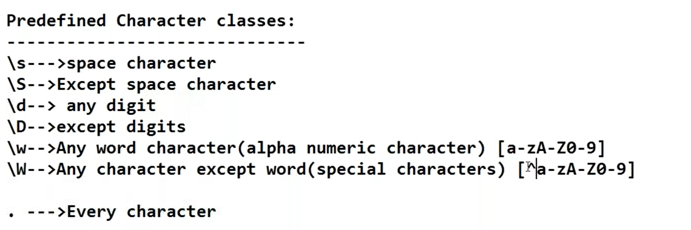
pattern = re.compile(‘python’)

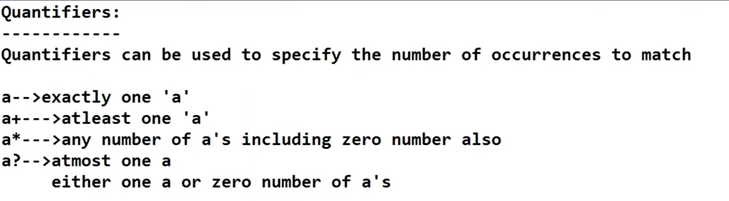
1. finditer ():

matcher = pattern.finditer(“ Learning Python is very easy”)

1. start (): start index of the match
2. end (): end+1 index of the match
3. group() : Returns matched String

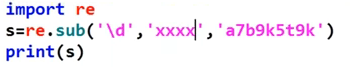








Substitution:



Important Function:

