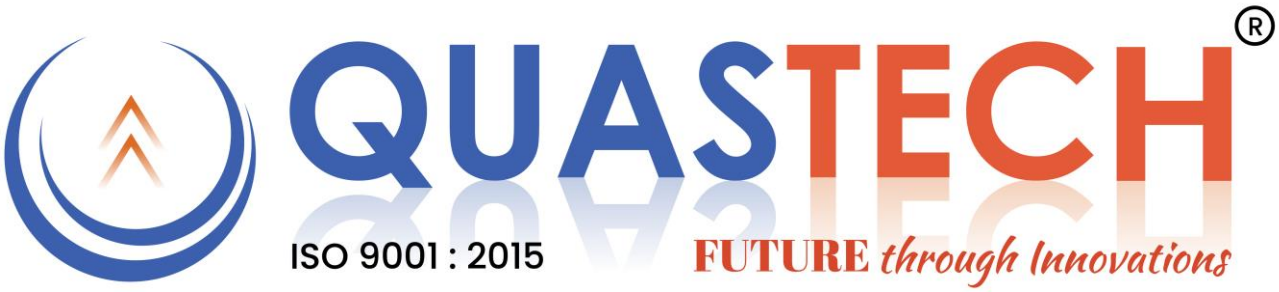


QUORATE SOFTWARE SOLUTIONS AND TECHNOLOGIES

Python For Data Analytics
Version 1.0

www.quastech.in

Head Office Address
201, Anant Laxmi Chambers,
Opp Waman Hari Pethe Jewellers,
Dada Patil Wadi Road,
Thane (W),
Maharashtra 400602
Contact No. - +91-8422800381 / 82 / 83

Borivali Branch
A/401, Court Chamber, Opp. Moksh Plaza,
S.V. Road, Borivali (W) - 400092
enquiry.borivali@quastech.in
Contact No. - +91 8422800384

Mohali Branch
SCF 62, Third Floor, Phase 7, Sector 61,
Sahibzada Ajit Singh Nagar,
Mohali, Punjab 160062
enquiry.mohali@quastech.in
Contact No. - +91 7208008461

Table of Contents

Python Introduction:

Why learn Python?
Python is easy language like english if you know any programming language then it would be beneficial but even though you dont know any programming language then that is also not aproblem. We are learning Python from scratch so no need to worry.Anyone can learn Python. If you want to learn any programming language then it is best suited for you.

Biggest advantage of Python is that we can write code very easily just like english statement.It uses less code than any other programming language.

What is Python?

- Python is general purpose high level programming language.
- It can be used to develop any kind of application like desktop, web, data science, machine learning.
- It is easy to understand for normal person because it is like english language.

Who developed Python?

Guido Van Rossam

When Python was developed?

- In 1989 while working at National Research Institute(NRI) in Netherland
- But Python was officially made available to public in 1991 (February 20th 1991)

Why the word Python was chosen?

- Because it can move through anything.
- It can bite any other language
- True -> there was one show Monty Python‘s circus in late 1969 to 1974

For Hadoop there is a symbol of yellow cap elephant, why?
Its inventor was thinking of some logo to define Hadoop. At that time his child was playingwith yellow cap elephant then he thought why not keep it as sign. That's the reason why Hadoop symbol in yellow cap elephant.

Features provided by Python ?

1. Functional Programming language – C
2. Object Oriented Programming feature – C++
3. Scripting Languages – Perl ,Shell script
4. Modular Programming Features – Modula-3Modular ? Programming divided in modules

Note: Most of the syntax is borrowed from C and ABC language

Where we can use Python?

A) Desktop/Standalone Application – e.g. calculator, system based application
B) Web Application
C) Database Application
D) Networking Application
E) Games – most common used programming language
F) Data Analysis – data science
G) Machine Learning Application
H) AI Application
I) IOT Application

For mobile based application Python is not recommended

Which companies are using Python?

- Google – search algorithm
- Facebook
- YouTube
- Dropbox – internally using Python
- NASA
- NSE – National Stock Exchange

Features of Python

1. Simple and easy to learn.
2. Free ware (license free) and Open source (Able to see source code).
3. High level programming language.
4. Platform Independent (WORA- write once run anywhere).
5. Portability – eg. mobile number portability (moving Python program from windows to Linuxmachine) – migrating one platform to another platform.
6. Dynamically typed.
7. Both procedure oriented and object oriented.
8. Interpreted.
9. Extensible – we can use other language programs in Python (we can use already existing code).
10. Embedded – use Python program with any other languages.
11. Extensive Library

Limitations of Python

- Performance
- Not used for mobile application

Flavors of Python

- CPython – standard flavor of Python used to worked with c language applications
- JPython or Jython – It is for java application
- IronPython – It is for dot net, C#
- PyPy – Python for speed (If want more performance then go for this Python) – Work with Python virtual machine i.e. JIT(just in time) compiler
- RubyPython – used with Ruby platform
- AnacondaPython – to handle huge data we can use this
- Stackless – (Python for accuracy i.e. multithreading)

Python versions?

- 1.0 – jan 1994
- 2.0 – oct 2000
- 3.0 – dec 2008

First Program in Python

Java Code for simple hello world

```
public class HelloWorld {
public static void main(String[] args)
{ System.out.println("Hello World");
}
}
```

Save file : HelloWorld.java
Comple java file : javac HelloWorldRun class file : java HelloWorld

Same code in C

```
#include<stdio.h>void main(){
printf("Hello World");
}
```

Simple Python Program

File : Hello.py

```
print("hello")
```

Output:
Hello

Comment in Python

 # comment - single line comment

''' comment ''' or """ comment """ - multiple line comment

Note: in java also we can do this with single statement but it came later i.e. in java 1.9 sept. 21St2017

Question : if you want to some of two number how you can do in java, c and Python.

Note: C, Java are static programming language that means in beginning only you need to assigntype of variable. In Python based on your provided value type of variable is decided

Variables

Variable is a name which is used to refer memory location. Variable also known asidentifier and used to hold value.

What is identifiers

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore or digit (0-9).

- Identifier name must not contain any white-space or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive for example my name, and MyName is not the same.
- Examples of valid identifiers : a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

Example:
Please refer to the Trainers classroom examples:

Python Operators

The operator can be defined as a symbol which is responsible for a particular operation between twooperands

- Arithmetic operator

Addition (+) Subtraction (-) Divide (/) Multiplication (*)Remainder (%) Exponent (**) Floor Division (//)

- **Comparison Operator / Relatonal Operator**

Double equal (==)
Less than or equal (<=) Greater than or equal (>=)Not equal (!=) Less than (<) Greater than (>)

- Assignment Operators

= , +=, -=, /=, *=, %=, **=, //=

- **Logical Operators**

and or not

- **Membership Operators**

in not in

- Identity Operators

is is not

- ***Slice operator* or slicing :** applicable for string and any sequwnce type

-6	-5	-4	-3	-2	-1
1	6	8	9	98	32
0	1	2	3	4	5

Access index in negative and positive onlythis is applicable for string only

syntax: sequence[begin:end:step] #all are optional

Note: begin value should be lesser than end value and vice versa. There is no index type of error forslice operator.

Python Data types

Python is a dynamically typed language hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type. Python provides us the **type()**function which returns the type of the variable passed.

Standard Data Types in Python

Numbers ***int(9)float*** ***complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)***
String
Boolean
List
Tuple
Dictionary (Key Value Pair Data)
Set

Example:
Please refer to the Trainers classroom examples:

Keywords in Python:

Key words for flow control / loops: if , else , elif , break continue ,return , while , for in , is , not
Key words for Function: def
Key words for class:
class
Key words for exception: try, finally,except
Key words for importing file: import,from , as
Other key words: True , False , None , and , or , asset , global nonlocal, raise, pass del, lambda

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to anotherdata type is called type conversion. Python has two types of type conversion.

1. Implicit Type Conversion
2. Explicit Type Conversion

Implicit Type Conversion:

In Implicit type conversion, Python automatically converts one data type to another data type.This process doesn't need any user involvement.

Let's see an example where Python promotes conversion of lower datatype (integer) to higher datatype (float) to avoid data loss.

```
num_int = 123
num_flo = 1.23num_new = num_int + num_floprint("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo)) print("Value of num_new:",num_new) print("datatype of num_new:",type(num_new))

Output:
datatype of num_int: <class 'int'> datatype of num_flo: <class 'float'>
Value of num_new: 124.23
datatype of num_new: <class 'float'>
```

Explicit Type Conversion:

In Explicit Type Conversion, users convert the data type of an object to required data type. Weuse the predefined functions like int(), float(), str(), etc to perform explicit type conversion.

This type conversion is also called typecasting because the user casts (change) the data type of theobjects. Typecasting can be done by assigning the required data type function to the expression.

```
num_int = 123
num_str = "456"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))num_sum = num_int + num_str
print("Sum of num_int and num_str:",num_sum)print("Data type of the sum:",type(num_sum))

Output:
Data type of num_int: <class 'int'>
Data type of num_str before Type Casting: <class 'str'>Data type of num_str after Type Casting:
<class 'int'> Sum of num_int and num_str: 579
Data type of the sum: <class 'int'>
```

Note:

1. Type Conversion is the conversion of object from one data type to another data type.
2. Implicit Type Conversion is automatically performed by the Python interpreter.
3. Python avoids the loss of data in Implicit Type Conversion.4. Explicit Type Conversion isalso called Type Casting, the data types of object are converted

using predefined function by user.

5. In Type Casting loss of data may occur as we enforce the object to specific data type.

Type conversion methods:

- int()
- float()
- chr()
- str()
- complex()
- bool()

Python control statements

It is required when we want to execute a code only if a certain condition is satisfied.
The if...elif...else statement is used in Python for decision making.Python if Statement Flowchart

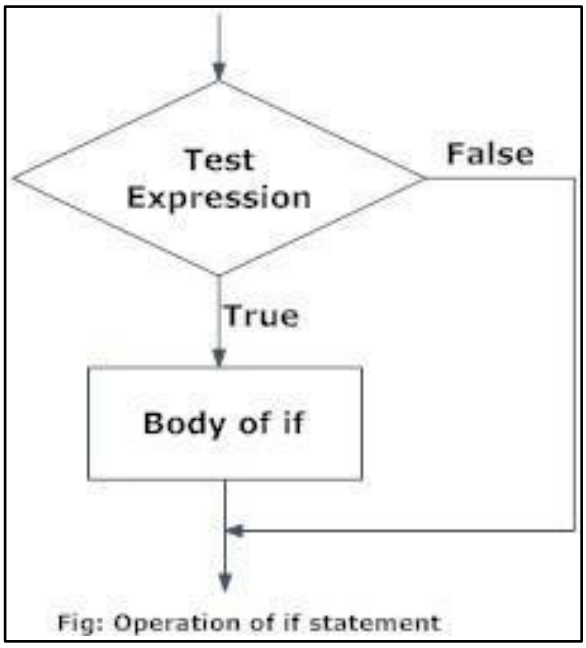


Fig: Operation of if statement

Python if..else Flowchart

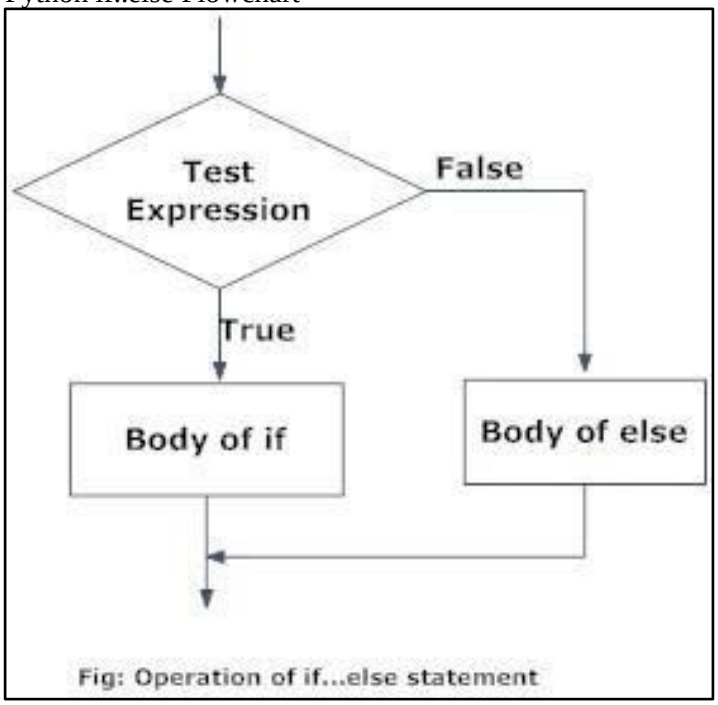


Fig: Operation of if...else statement

Flowchart of if...elif...else

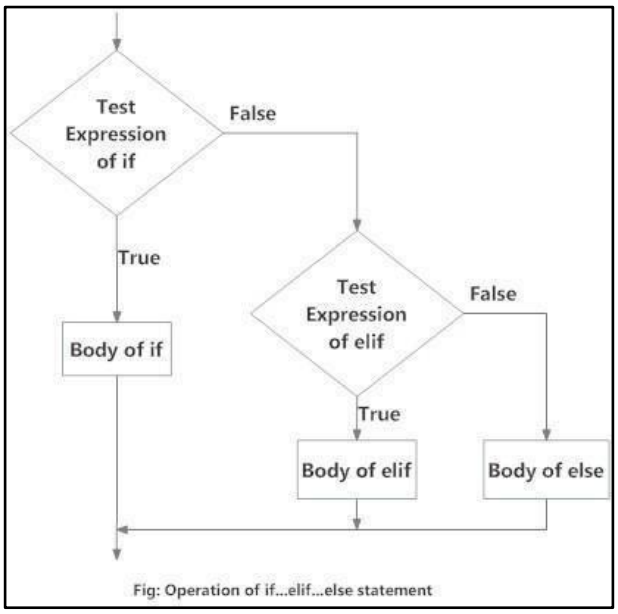


Fig: Operation of if...elif...else statement

Syntax:

if:

if condition:
true case statement

if else :

if condition:
true case statementelse:
false case statement

else if :

if condition:statement
elif condition:statement

Example:
Please refer to the Trainers classroom examples:

Python Loop

For Loop

The for loop in Python is used to iterate over a sequence (list , tuple, string) or other iterableobjects. Iterating over a sequence is called traversal.

Syntax:

```
for val in sequence:
      Body of for
```

Here, val is the variable that takes the value of the item inside the sequence on each iteration.

Loop continues until we reach the last item in the sequence. The body of for loop is separated from the rest of the code using indentation.

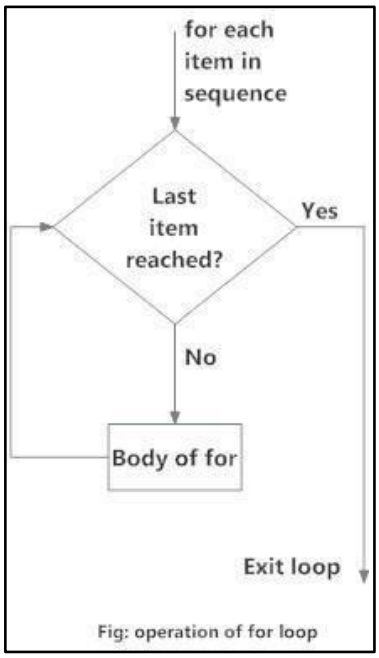


Fig: operation of for loop

The range() function

We can generate a sequence of numbers using range() function. range(10) will generatenumbers from 0 to 9 (10 numbers).

We can also define the start, stop and step size as range(start,stop,step size). step size defaults to 1 ifnot provided.

for loop with else

A for loop can have an optional else block as well. The else part is executed if the items in thesequence used in for loop exhausts.

break statement can be used to stop a for loop. In such case, the else part is ignored.Hence, a for loop's else part runs if no break occurs.

Syntax:

```
for loopvariable in range/list:
    statement
```

- range function parameters - range(start, stop, step)
- by default start is 0 and step i.e. increment or decrement is 1
- list is normal list

Example of loops:

For and while loop :

1)

```
for i in range(5):
    print(i)
```

output:

```
0
1
2
3
4
```

While loop

The while loop in Python is used to iterate over a block of code as long as the test expression(condition) is true. We generally use this loop when we don't know beforehand, the number of times to iterate.Syntax:

```
while test_expression:
Body of while
```

In while loop, test expression is checked first. The body of the loop is entered only if the test_expression evaluates to True. After one iteration, the test expression is checked again. Thisprocess continues until the test_expression evaluates to False. Python interprets any non-zero valueas True. None and 0 are interpreted as False.

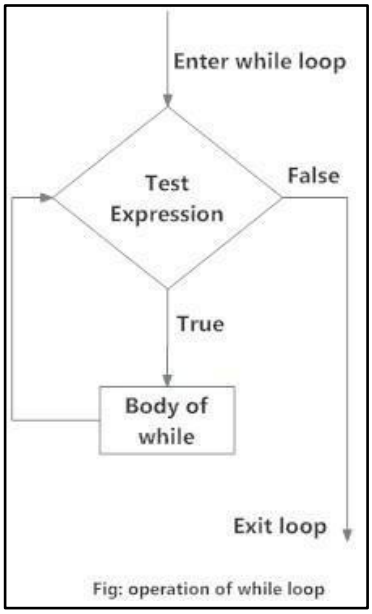


Fig: operation of while loop

Note: We need to increase the value of counter variable in the body of the loop. This is very important (and mostly forgotten). Failing to do so will result in an inf__init__e loop (never ending loop).**while loop with else** Same as that of for loop, we can have an optional else block with while loop as well.The else part isexecuted if the condition in the while loop evaluates to False.

The while loop can be terminated with a break statement. In such case, the else part is ignored.Hence, a while loop's else part runs if no break occurs and the condition is false.

```
i = 0
while i < 5:
  print(i, end=" ")
  i += 1

Output: 0 1 2 3 4
```

Exercise1:

I. WAP to check whether give number is prime or not
II. WAP to check palindrome for string and number both
III. WAP to check given value(string or number) is armstrong or not
IV. WAP of fibonacci series
V. Swap two variables without using third variable
VI. Check for even or odd without using % and /
VII. Swap three variables without using fourth variable

What is the use of break and continue in Python?

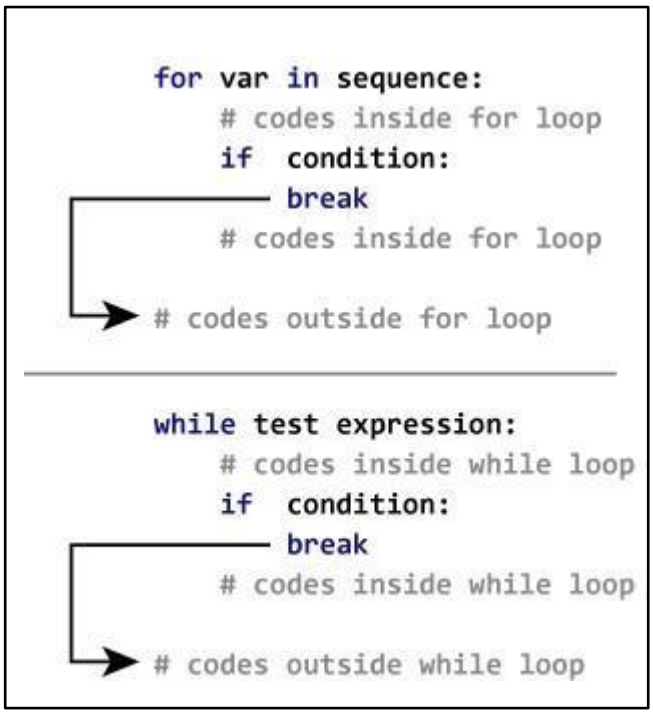
In Python, break and continue statements can alter the flow of a normal loop.
Loops iterate over a block of code until test expression is false, but sometimes we wish to terminatethe current iteration or even the whole loop without checking test expression.
The break and continue statements are used in these cases.
Python break statement
The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.
If break statement is inside a nested loop (loop inside another loop), break will terminate theinnermost loop.

Syntax of break

```
break
```

Flowchart of break

The working of break statement in for loop and while loop is shown below.



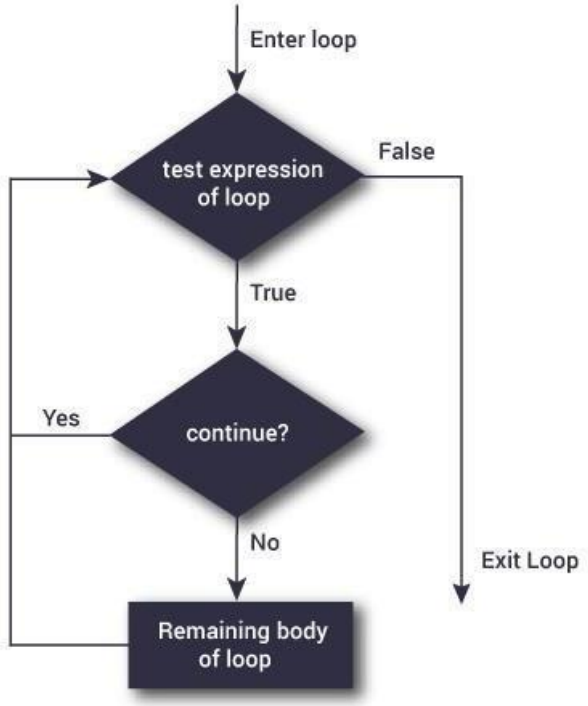
Example:
Please refer to the Trainers classroom examples:

Python continue statement

The continue statement is used to skip the rest of the code inside a loop for the current iterationonly. Loop does not terminate but continues on with the next iteration.
Syntax of Continue

```
continue
```

Flowchart of continue



The working of continue statement in for and while loop is shown below.
Example:
Please refer to the Trainers classroom examples:

What is pass statement in Python?

In Python programming, pass is a null statement. The difference between a comment and pass statement in Python is that, while the interpreter ignores a comment entirely, pass is not ignored.However, nothing happens when pass is executed. It results into no operation (NOP).
Syntax of pass

```
pass
```

Example
Please refer to the Trainers classroom examples:

We generally use it as a placeholder.
Suppose we have a loop or a function that is not implemented yet, but we want to implement it inthe future. They cannot have an empty body. The interpreter would complain. So, we use the passstatement to construct a body that does nothing.

Python String Functions:

A string is just a sequence of characters. It is among the most popular **data types in Python**. It can be created simply by enclosing characters in quotes. Python provides a rich set of operators, functions, and methods for working with strings, which helps access and extract portions of strings and manipulate and modify string data.

str = **'helloPython''''**

Syntax:

capitalize()

- Only its first character capitalized.

```
print(str.capitalize())
```

Syntax:

str.count(sub, start= 0,end=len(string))

number of occurrences of substring sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

```
print(str.count('h', 1, len(str)))
```

Syntax:

str.endswith(suffix[, start[, end]]) str.startswith(suffix[, start[, end]])return true false value result.

suffix − This could be a string or could also be a tuple of suffixes to look for.start − The slice begins from here.

end − The slice ends here.

```
check = 'he' print(str.endswith(check)) print(str.startswith(check))
```

Syntax:

str.find(str, beg=0, end=len(string))

It determines if string str occurs in string str − This specifies the string to be searched.

beg − This is the starting index, by default its 0.

end − This is the ending index, by default its equal to the length of the string.

```
print(str.find('me'))
```

Syntax:

str.index(str, beg = 0 end = len(string))

Returns index if found otherwise raises an exception if str is not found.str − This specifies the string to be searched.

beg − This is the starting index, by default its 0.

end − This is the ending index, by default its equal to the length of the string.

```
print(str.index('n'))
```

Syntax:

str.isalpha()

check for if string is alphabet or notstr.isdigit()

check for if string is digit or notstr.isdecimal()

check for if string is decimal or notstr.isupper()

check for if string is in uppercase or notstr.islower()

check for if string is in lowecase or notstr.isspace()

check for only spacesstr.isalnum()

check for alpha, numeric or alpha numeric valuesstr.isnumeric()
Returns true if a unicode string contains only numeric characters and false otherwise.

convert string to uppercasestr.lower()
conver string to lowercaselen(str)

```
print('isalpha : ', str.isalpha())
print(str.upper())

print('isdigit : ', str.isdigit()) print('isdecimal : ', str.isdecimal()) print('isupper : ', str.isupper())
print('islower : ', str.islower()) print('isspace : ', str.isspace()) print('isnumeric : ', str.isnumeric())
print('isalnum : ', str.isalnum())

print('upper : ', str.upper())
print('lower : ', str.lower())
print('len(str)', len(str))
```

Syntax:
str.join(sequence)
returns a string in which the string elements of sequence have been joined by str separator.

```
s = "-"
seq = ("a", "b", "c")
print(s.join(seq))
```

Syntax:
str.lstrip([char])
Removes all leading whitespace in string.str.rstrip([char])
Removes all trailing whitespace in string.str.strip([char])
perform lstrip and rstrip both.

```
newstr = '        Hello ' print(newstr) print(newstr.lstrip()) print(len(newstr)) newstr = newstr.rstrip()
print(len(newstr))
```

Syntax:
str.split(str="", num=string.count(str)).
str − This is any delimeter, by default it is space.num − this is number of lines minus one
str.splitlines()

```
newstr = 'hello hi bye '
print(newstr)
str3 = newstr.split()print(str3)
```

Syntax:
string.replace(oldvalue, newvalue, count)

oldvalue Required. The string to search for
newvalue Required. The string to replace the old value with
count Optional. A number specifying how many occurrences of the old value you want toreplace. Default is all occurrences

```
txt = "one one was a race horse, two two was one too."
x = txt.replace("one", "three")print(x)
```

Syntax:

string.rfind(value, start, end)

Searches the string for a specified value and returns the last position of where it was found

string.rindex(value, start, end)

Searches the string for a specified value and returns the last position of where it was found

'''

```
txt = "Hello, welcome to my world."
x = txt.rfind("e")print(x)
x2 = txt.rindex("e")print(x2)
```

List

A list refers to a collection of objects; it represents an ordered sequence of data. In that sense, a listis similar to a string, except a string can hold only characters. We may access the elements contained in a list via their position within the list. A list need not be homogeneous; that is, the elements of a list do not all have to be of the same type**.**

List is sequence of homogenous or non-homogenous data. It is __init__ialized using square ([ ]) brackets

Example :

```
ls = [] #empty list
ls2 = [ – Rahul , –Virat ] # homogenous data
ls3 = [1, –Nandini, 15000.0] # non - homogenous data
```

The number within the square brackets indicates the distance from the beginning of the list. The expression list[0] therefore indicates the element at the very beginning (a distance of zero from thebeginning), and list[1] is the second element (a distance of one away from the beginning).

If a is a list with n elements, and i is an integer such that 0 ≤ i <n, then a[n] is an element in the list.Lst = [5, -3, 12] # homogenous data

Simple list with three elements. The small number below a list element represents the index of thatelement.

The expression within [] must evaluate to an integer; some examples include

- an integer literal: a[34]
- an integer variable: a[x] (x must be an integer)
- an integer arithmetic expression: a[x + 3] (x must be an integer)
- an integer result of a function call that returns an integer: a[max(x, y)] (max must return aninteger)
- an element of another list: a[b[3]] (element b[3] must be an integer)

The action of moving through a list visiting each element is known as traversal. The for loop ismade to iterate over aggregate types like lists.

The statement

```
a = [2, 4, 6, 8]
```

assigns the given list literal to the variable a. The expression

```
a + [1, 3, 5]
```

evaluates to the list [2, 4, 6, 8, 1, 3, 5], but the statement does not change the list to which a refers.

The statement

```
a = a + [1, 3, 5]
```

actually reassigns a to the new list [2, 4, 6, 8, 1, 3, 5]. The statement

```
a += [10]
```

updates a to be the new list [2, 4, 6, 8, 1, 3, 5, 10]. Observe that the + will concatenate two lists,but it cannot join a list and a non-list. The following statement

```
a += 20
```

is illegal since a refers to a list, and 20 is an integer, not a list. If used within a program under theseconditions, this statement will produce a run-time exception.

Slicing in list :

We can make a new list from a portion of an existing list using a technique known as slicing. A listslice is an expression of the form

list [begin : end]

Where • list is a variable referring to a list object, a literal list, or some other expression thatevaluates to a list,

- begin is an integer representing the starting index of a subsequence of the list, and
- end is an integer that is one larger than the index of the last element in a subsequence of the list.

If missing, the begin value defaults to 0. A begin value less than zero is treated as zero. If the endvalue is missing, it defaults to the length of the list. An end value greater than the length of the list is treatedas the length of the list.

list[start:stop:step]

Example:

```
ls2 = [1, "hello", 3, 4.23, 5]

# from index/position 1 to less than 4th index increment by 2 positionprint(ls2[1:4:2])

# all elementsprint(ls2[:])

# last to first position - reverse listprint(ls2[::-1])

# index 2nd to less than 4 increment by 1 positionprint(ls2[2:4:1])

# 4th index to last increment by twoprint(ls2[4::2])

# all elementsprint(ls2[::])
```

Output :

```
['hello', 4.23]
[1, 'hello', 3, 4.23, 5]
[5, 4.23, 3, 'hello', 1]
[3, 4.23]
[5]
[1, 'hello', 3, 4.23, 5]
```

Python List Operator :

- Repetition (*)
- Concatenation (+)
- Membership (in, not in)
- Iteration (using for loop)
- length (len(list) function)

Example:

Please refer to the Trainers classroom examples:

Python List Built-in functions

1) Add element in list
- **list.insert(index,object):** The object is inserted into the list at the specified index. Objectcan be list type or single element.
- **list.append(object):** The element represented by the object is added to the list at last indexas element of list. Object can be list type or single element.
- **list.extend(sequence):** The sequence represented by any type of sequance which is iterable.

Example of each method:
list.insert(index,object)
Please refer to the Trainers classroom examples:

list.append(object)
Please refer to the Trainers classroom examples:

list.extend(sequence) - sequence is iterable type parameter
Please refer to the Trainers classroom examples:

2) Get element of list and index of element in list
- **list.index(object):** It return index of object passed as parameter, if object doesnt exist in list then give ValueError at runtime.
- **list[start:end:step]:** Use slicing to get element of list.

3) Delete/Remove element from list
- **list.clear():** remove all elements of list.
- **del list:** It deletes list and if we try to access deleted list then NameError is given as output.It works with any type of variable
- **list.remove(object):** It removes the specified object from the list. If duplicate object is there in list then first matching object is deleted. If object is not found then ValueError is thrown.
- **list.pop(index=list[-1]):** remove last element of list by default if index is not specified. If index is specified then element of that index is removed from list. If index is not available then IndexError is thrown.

Example:
4) Sort list
- **list.sort(reverse=False):** Sort the element of list and by default it sorts list in ascending order. It return **None**. To sort elements in descending specify **reverse=True** as parameter in sort method. It is compusory to have homogenous list to use this method.

5) Reverse list
- **list.reverse():** It reverses the list.

6) Count ocurrance of specified element in list
- **list.count(object):** It returns the number of occurrences of the specified object in the list. Object can be element or sequance.

7) Create Copy of list
- **list.copy():** It returns a shallow copy of the list.

list Methods
- ✓ **count :** Returns the number of times a given element appears in the list. Does not modify the list.**Insert :** Inserts a new element before the element at a given index. Increases the length of the listby one. Modifies the list.
- ✓ **Append :** Adds a new element to the end of the list. Modifies the list.
- ✓ **Index :** Returns the lowest index of a given element within the list. Produces an error if the elementdoes not appear in the list. Does not modify the list.

- ✓ **Remove :** Removes the first occurrence (lowest index) of a given element from the list. Produces anerror if the element is not found. Modifies the list if the item to remove is in the list.
- ✓ **Reverse :** Physically reverses the elements in the list. The list is modified.
- ✓ **Sort :** Sorts the elements of the list in ascending order. The list is modified.
- ✓ ***Some other function of list: obj.min(sequence), obj.max(sequence)***

```
ls = [1, 3, 5, 1, 5, 1]
print("minimum = ", min(ls)) print("maximum = ", max(ls)) print("minimum = ", min([23, 54, 67, 12, 3]))
print("maximum = ", max([23, 54, 67, 12, 3]))

Output:
minimum =  1
maximum =  5
minimum = 3
maximum = 67
```

Convert any type to list: list(sequence)

List Comprehension

List comprehensions are used for creating new **lists** from other iterables. As **list comprehensions** returns **lists** , they consist of brackets containing the expression, which is executedfor each element along with the for loop to iterate over each element.

```
l = [x for x in range(10)]print(l)
output:
[0,1,2,3,4,5,6,7,8,9]
```

Tuple

Tuple is set of homogeneous and non-homogeneous data. Tuple is declared using round () bracktes.

Example:

ls = ()

ls = (1, –Nandini|, 12000.0)

ls2 = (22, 43, 56, 23)

Python Tuple Operator :

The operators like concatenation (+),repetition (*),

Membership (in) works in the same way as they work with the list.

Tuple inbuilt functions:

len(tuple): Length of tuple **index(object):** position of object **count(object):** occurrence of object **max(tuple):** Maximum of tuple(data should be homogeneous)**min(tuple):** Minimum of tuple(data should be homogeneous) **tuple(sequence):** Convert sequence to tuple type

Where use tuple

Using tuple instead of list is used in the following scenario:

1. Using tuple instead of list gives us a clear idea that tuple data is constant and must notbe changed.

2. Tuples that contain immutable elements can be used as a key for a dictionary. With lists, thisis not possible.
   t = (('m', 8), ('n', 9))
   d = dict(t, o=10) output: {_m‘: 8, _n‘: 9, _o‘: 10}

3. Tuple can be used as the key inside dictionary due to its immutablenature. d = { (1,'ajay'):'hello' }
   print(d[(1,'ajay')])Output: hello

List VS Tuple

List	Tuple
The literal syntax of list is shown by the [].	The literal syntax of the tuple is shown by the ().
The List is mutable.	The tuple is immutable.
The List has the variable length.	The tuple has the fixed length.
The list provides more functionality than tuple.	The tuple provides less functionality than the list.
The list Is used in the scenario in which we need to store the simple collections with no constraints where the value of the items can be changed.	The tuple is used in the cases where we need to store the read-only collections i.e., the value of the items cannot be changed. It can be used as the key inside the dictionary.

Nesting List and tuple

We can store list inside tuple or tuple inside the list up to any number of level.Lets see an example of how can we store the tuple inside the list.

Python Dictionary

Dictionary is used to implement the key-value pair in Python. The dictionary is the data type in Python which can simulate the real-life data arrangement where some specific value exists for someparticular key.

Example : Dict = {"Name": "Ayush","Age": 22}

Built-in Dictionary functions

The built-in Python dictionary methods along with the description are given below.

Function Description

len(dict) It is used to calculate the length of the dictionary.

str(dict) It converts the dictionary into the printable string representation.type(variable) It is used to print the type of the passed variable.

Built-in Dictionary methods

The built-in Python dictionary methods along with the description are given below.

Method	Description
dic.clear()	It is used to delete all the items of thedictionary.
dict.copy()	It returns a shallow copy of the dictionary.
dict.fromkeys(iterable, value = None, /)	Create a new dictionary from the iterablewith the values equal to value.
dict.get(key, default = "None")	It is used to get the value specified for the
	passed key.
dict.items()	It returns all the key-value pairs as a tuple.
dict.keys()	It returns all the keys of the dictionary.
dict.setdefault(key,default= "None")	It is used to set the key to the default valueif the key is not specified in the dictionary
dict.update(dict2)	It updates the dictionary by adding the key-value pair of dict2 to this dictionary.
dict.values()	It returns all the values of the dictionary.
dict.popitem()	Removes first item in the dictionary.
dict.pop()	Removes specified item from dictionary, ifkey is not present then gives KeyError

Example:

Please refer to the Trainers classroom examples:

Dictionary Comprehension

Dictionary comprehensions are used for creating new **dictionary** from other iterables. As **dictionary comprehensions** returns **dictionary,** they consist of brackets containing the expression,which is executed for each element along with the for loop to iterate over each element.

```
l = {x : x for x in range(10)}print(l)
output:
{0:1, 1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8, 9:9}
```

Set

The set in Python can be defined as the unordered collection of various items enclosed within the curly({}) braces. The elements of the set cannot be duplicate. The elements of the Python set mustbe immutable.

Example 1: using curly braces

Please refer to the Trainers classroom examples:

Example 2: using set() method

Python Set operations

However, we can perform various mathematical operations on Python sets like union,intersection, difference, etc.

Adding items to the set

Python provides the add() method which can be used to add some particular item to the set.

Consider the following example.

To add more than one item in the set, Python provides the **update()** method.

Removing items from the set

Python provides **discard()** method which can be used to remove the items from the set.

Python also provide the **remove()** method to remove the items from the set. Consider the followingexample to remove the items using remove() method.

We can also use the **pop()** method to remove the item. However, this method will always removethe last item.

Python provides the **clear()** method to remove all the items from the set.

Difference between discard() and remove()

Despite the fact that discard() and remove() method both perform the same task, There is one main difference between discard() and remove(). *If the key to be deleted from the set using discard() doesn't exist in the set, the Python will not give the error. The program maintains its control flow. On the other hand, if the item to be deleted from the set using remove() doesn't exist in the set, the Python will give the error.*

Union of two Sets

The union of two sets are calculated by using the or (|) operator. The union of the two setscontains the all the items that are present in both the sets.

Consider the following example to calculate the union of two sets.

Example 1: using | operator Example 2: using union() methodPlease refer to the Trainers classroom examples:

Intersection of two sets

The & (intersection) operator is used to calculate the intersection of the two sets in Python.

The intersection of the two sets are given as the set of the elements that common in both sets.

Example 1: using & operator

Example 2: using intersection() method

The intersection_update() method

The intersection_update() method removes the items from the original set that are not present inboth the sets (all the sets if more than one are specified).

The Intersection_update() method is different from intersection() method since *it modifies the original set by removing the unwanted items, on the other hand, intersection() method returns anew set.*

Example:

Please refer to the Trainers classroom examples:

Difference of two sets

The difference of two sets can be calculated by using the *subtraction (-)* operator. The resulting set will be obtained by *removing all the elements from set 1 that are present in set 2*.**Example 1:**

Example 2 : using difference() method

Please refer to the Trainers classroom examples:

Set comparisons

Python allows us to use the *comparison operators i.e.*, <, >, <=, >= , == with the sets by using which we can *check whether a set is subset, superset, or equivalent to other set*. The booleantrue or false is returned depending upon the items present inside the sets.

```
a = {"ayush", "biba","babu"}b = {"ayush", "biba"}
c = {"ayush", "biba", "babu", "isha"}print("a > b : ", (a > b))
print("a > c : ", (a > c))
print("b == c : ", (b == c))

Output:
a > b : Truea > c : False
b == c : False
```

Frozen Set

The frozen sets are the immutable form of the normal sets, i.e., the items of the frozen setcan not be changed and therefore it can be used as a key in dictionary.

Example:

Please refer to the Trainers classroom examples:

Frozenset for the dictionary

If we pass the dictionary as the sequence inside the frozenset() method, it will take only the keysfrom the dictionary and returns a frozenset that contains the key of the dictionary as its elements. ***Example:***

Please refer to the Trainers classroom examples:

Python Functions

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Defining a Function

Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses.
- Any input parameters or arguments should be placed within these parentheses. You canalso define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string ofthe function or *docstring.*
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression tothe caller. A return statement with no arguments is the same as return None.

Code is made more reusable by packaging it in functions. A function is a unit of reusable code. we will see how to write our own reusable functions, but in this chapter we examine some of the functions available in the Python standard library. Python provides a collection of standard code stored in libraries called modules. Programmers can use parts of this librarycode within their own code to build sophisticated programs.

Syntax

```
def functionname( parameters ):"function_docstring" function_suite
return [expression]
```

Example

```
def printme( str ):print(str)
return
```

Calling a function

```
def myfun():
print(—hello‖) #calling myfun functionmyfun()
```

```
Output:hello
```

Parameters in function

The information into the functions can be passed as the parameters. The parameters are specified in the parentheses. We can give any number of parameters, but we have to separate themwith a comma.

Example 1:

```
# defining the functiondef func(name):
print("Hi ", name)# calling the functionfunc("Ayush")

Output:
Hi Ayush
```

Example 2:

```
# Python function to calculate the sum of two variables# defining the function
def sum(a, b):return a + b
# taking values from the usera = int(input("Enter a: "))
b = int(input("Enter b: ")) # printing the sum of a and bprint("Sum = ", sum(a, b))
```

```
Output:
Enter a: 5
Enter b: 8
Sum =  13
```

Call by reference in Python

In Python, all the functions are called by reference, i.e., all the changes made to the referenceinside the function revert back to the original value referred by the reference.
However, there is an exception in the case of immutable objects since the changes made to the immutable objects like string do not revert to the original string rather, a new string object is made,and therefore the two different objects are printed.

Example 1 Passing mutable Object (List)

```
# defining the functiondef change_list(list1):
list1.append(20) list1.append(30)
print("list inside function = ", list1)# defining the list
list1 = [10, 30, 40, 50]
# calling the functionchange_list(list1)
print("list outside function = ", list1)

Output:
list inside function = [10, 30, 40, 50, 20, 30]
list outside function = [10, 30, 40, 50, 20, 30]
```

Example 2 Passing immutable Object (String)

```
# defining the functiondef change_string(str):
str = str + " Howare you"
print("printing the string inside function :", str)string1 = "Hey there"
# calling the function change_string(string1)
print("printing the string outside function :", string1)

Output:
printing the string inside function : Hey there How are youprinting the string outside function : Hey
there
```

Recursion

The factorial function is widely used in combinatorial analysis (counting theory in mathematics),probability theory, and statistics. The factorial of n usually is expressed as n!. Factorial is defined for non-negative integers as $n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots 3 \cdot 2 \cdot 1$and 0! is defined to be 1. Thus $6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 =$
720.

```
def factorial(n):if n == 0:
return 1

else:
return n * factorial(n - 1)

def main():
print(factorial(0))print(factorial(6))print(factorial(10)
)
```

main()

Types of arguments/parameters

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

Required Arguments

As far as the required arguments are concerned, these are the arguments which are requiredto be passed at the time of function calling with the exact match of their positions in the function call and function def__init__ion. If either of the arguments is not provided in the function call, or the position of the arguments is changed, then the Python interpreter will show the error.

Example:

Please refer to the Trainers classroom examples:

Keyword arguments

This kind of function call will enable us to pass the arguments in the random order. Thename of the arguments is treated as the keywords and matched in the function calling and def__init__ion. If the same match is found, the values of the arguments are copied in the functiondef__init__ion.

Example

Please refer to the Trainers classroom examples:

If we provide the different name of arguments at the time of function call, an error will be thrown.

Example

Please refer to the Trainers classroom examples:

Note: The Python allows us to provide the mix of the required arguments and keyword arguments atthe time of function call. However, the required argument must not be given after the keyword argument, i.e., once the keyword argument is encountered in the function call, the following arguments must also be the keyword arguments.

Example

Please refer to the Trainers classroom examples:

The following example will cause an error due to an in-proper mix of keyword and requiredarguments being passed in the function call.

Example

Please refer to the Trainers classroom examples:

Default Arguments

Python allows us to __init__ialize the arguments at the function def__init__ion. If the value of anyof the argument is not provided at the time of function call, then that argument can be __init__ialized with the value given in the def__init__ion even if the argument is not specified at the function call.

Example

Please refer to the Trainers classroom examples:

Variable length Arguments

In the large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to provide the comma separated values which are internally treated as tuples at the function call. However, at the function def__init__ion, we have to define the variable with * (star) as *<variable - name > e.g. *argument.

Example

Please refer to the Trainers classroom examples:

Keyword Variable length arguments

The special syntax ****kwargs** in function def__init__ions in Python is used to pass a keyworded, variable-length argument list. We use the name kwargs with the double star.The reason is because the double star allows us to pass through keyword arguments (andany number of them).

1. A keyword argument is where you provide a name to the variable as you pass it into the function.
2. One can think of the kwargs as being a dictionary that maps each keyword to the value that we pass alongside it.
3. That is why when we iterate over the kwargs there doesn‘t seem to be any order in which they were printed out.

Example

Please refer to the Trainers classroom examples:

Scope of variables

The scopes of the variables depend upon the location where the variable is being declared. Thevariable declared in one part of the program may not be accessible to the other parts.
In Python, the variables are defined with the two types of scopes.

1. Global variables

Variables defined within functions are local variables. Local variables have some verydesirable properties:

- The memory required to store a local variable is used only when the variable is in scope.When the program execution leaves the scope of a local variable, the memory for that variable is freed up and can be used for a local variable in another function when thatfunction is invoked.
- The same variable name can be used in different functions without any conflict. The interpreter derives all of its information about a local variable used within a function from the def__init__ion of that variable within that function. If the interpreter attempts to execute a statement that uses a variable that has not been defined, the interpreter issues a run-time error. When executing code in one function the interpreter will not look for a variable def__init__ion in another function. Thus, there is no way a local variable in one function can interfere with a local variable declared in another function. A local variable is transitory, so its value is lost in between function invocations. Sometimes it is desirable to have a variablethat lives as long as the program is running; that is, until the main function completes.

In contrast to a local variable, a global variable is defined outside of all functions and is notlocal to any particular function. Any function can legally access and/or modify a global variable.

2. Local variables

The variable defined outside any function is known to have a global scope whereas the variabledefined inside a function is known to have a local scope.

Example

Please refer to the Trainers classroom examples:

global Keyword

In Python, global keyword allows you to modify the variable outside of the current scope. It is usedto create a global variable and make changes to the variable in a local context.

Rules of global Keyword

The basic rules for global keyword in Python are:

- When we create a variable inside a function, it‘s local by default.

- When we define a variable outside of a function, it‘s global by default. You don‘t have touse global keyword.
- We use global keyword to read and write a global variable inside a function.
- Use of global keyword outside a function has no effect

Documenting Functions and Modules

It is good practice to document a function‘s def__init__ion with information that aids programmers who may need to use or extend the function. The essential information includes:

- The purpose of the function. The function‘s purpose is not always evident merely from its name. This is especially true for functions that perform complex tasks. A few sentences explaining what the function does can be helpful.
- The role of each parameter. The parameter names are obvious from the def__init__ion, but the type andpurpose of a parameter may not be apparent merely from its name.
- The nature of the return value. While the function may do a number of interesting things as indicated in the function‘s purpose, what exactly does it return to the client? It is helpful to clarify exactly what value the function produces, if any.

Module

What's an module?

A module is a file consisting of Python code. A module can define functions, classes andvariables. A module can also include runnable code.

Why we use module?

To use any package in your code, you must first make it accessible. You have to import it.
You can't use anything in Python before it is defined. Some things are built in, for example the basic types (like int, float, etc) can be used whenever you want. But most things you will want todo will need a little more than that.

A package is just a directory tree with some Python files in it. Nothing magical. If you wantto tell Python that a certain directory is a package then create a file called ____init__ .py and just stickit in there.

Create a Module

To create a module just save the code you want in a file with the file extension .py:

```
def greeting(name): print("Hello, " + name)
```

Use a Module

Now we can use the module we just created, by using the import statement:

```
import mymodule

mymodule.greeting("Jonathan")
```

Variables in Module

The module can contain functions, as already described, but also variables of all types (arrays,dictionaries, objects etc):

```
person1 = { "name": "John","age": 36,
"country": "Norway"
}
```

Example
Import the module named mymodule, and access the person1 dictionary:

```
import mymodule

a = mymodule.person1["age"]print(a)
```

Re-naming a Module
You can create an alias when you import a module, by using the as keyword:Example

Create an alias for mymodule called mx:

```
import mymodule as mx

a = mx.person1["age"]print(a)
```

Built-in Modules
There are several built-in modules in Python, which you can import whenever you like.Example

Import and use the platform module:

```
import random

x = random.random()print(x)
```

Using the dir() Function
There is a built-in function to list all the function names (or variable names) in a module.The dir() function:
Example
List all the defined names belonging to the platform module:

```
import platform

x = dir(platform)print(x)
```

Import From Module
You can choose to import only parts from a module, by using the from keyword.Example

The module named mymodule has one function and one dictionary:

```
def greeting(name): print("Hello, " + name)

person1 = { "name": "John","age": 36,
"country": "Norway"
}
```

Example

Import only the person1 dictionary from the module:

```
from mymodule import person1

print (person1["age"])
```

Note: When importing using the from keyword, do not use the module name when referring toelements in the module. Example: person1["age"], **not** ~~mymodule.person1["age"]~~

Some built in modules
Random Module :used to generate random number
Functions of random module

Function Name	Description
random()	random module by default generate randomvalues from 0 to 1 but not 0 or 1 i.e. in between 0 to 1
randint(start,end)	to give any range from start to end
randrange(start,stop,step)	generate random range from start to end
uniform(start,end)	generate random floating point values from startto end
choice(list)	if you want to get random value from list then use choice(list) function of random moduleit uses 1 sample at a time
sample(list,k=value)	if you want to give sample more than 1 then use sample(list,k=value) function of moduleit will generate k sample values of list randomly
shuffle(list)	shuffle function to shuffle data of list
triangular(low, high, mode)	Return a random floating point number betweenlow and high, with the specified mode between those bounds.

Example
Please refer to the Trainers classroom examples:

Time module

Time generated __init__ially from 1st january 1970. Python has a module named time to handletime-related tasks. To use functions defined in the module, we need to import the module first.

The time package contains a number of functions that relate to time. We will consider two: clockand sleep. The clock function allows us measure the time of parts of a program‘s execution. The clock returnsa floating-point value representing elapsed time in seconds. On Unix-like systems (Linux and Mac OS ,X), clock returns the numbers of seconds elapsed since the program began executing. Under Microsoft Windows, clock returns the number of seconds since the first call to clock. In either case,with two calls to the clock function we can measure elapsed time.

Functions of time module:

Function Name	Description
time()	It prints date in seconds i.e. floating type value
ctime(time)	Convert floating type time value in readable format or understandable format. Return date in string format.
sleep(seconds)	Function suspends (delays) execution of the current thread for the given number of seconds.
localtime(time)	Print time and date in tuple format
strftime(format,tuple)	Change format of date and time