**Name some commonly used library in pythons?**

os

csv

subprocess

copy

requests

re

**How to use logging in python?**

**What are features of python?**

Python is a dynamic, high level, free open source and interpreted programming language

Free and open source

Object Oriented

Interpreted language

Large standard library

Dynamically typed language

**How to check size of file using fucntion**

In python’s os.path model there is function get\_size() which returns the file size in Bytes

os.path.getsize(path) --- to get the size for given path in bytes

**How to get size of directory in python**

No builtin method, use walk function to get file size of each file and sum that value

**What is use of walk() in os module**

It walks from from give path to each subdirectory/subdirectory.

It returns tuple of three variable.

path, dir, file=os.walk(path)

path – represents where it is

dir ---- gives all directory in current path as list form

file ---- gives all files in that path in list

**What is zip() function in Python**

Zip function take each argument from a iterables and create a new zip object after picking single object from each iterables.

z=zip(a,b,c..)

**Use zip function for fetch elements from three iterables and data from one iterable must come in reverse order.**

list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

list3=range(4)

for x, y,z in zip(list1, list2[::-1],list3):

    print(x, y, z)

If length of each object is not equal then result will have length of minimum of of them.

x=list(range(10))

y=list(range(5))

print(x)

print(y)

z=zip(x,y)

for each in z:

    print(each)

output

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4]

(0, 0)

(1, 1)

(2, 2)

(3, 3)

(4, 4)

**What is the difference between remove() function and del statement?**

remove() ----- delete/removes given value from list

del --------- it deletes/removes value from given index

**WAP to get the ASCII code of each character**

We can use ord() to get the ASCII value of any character.

ord(‘e’) // 101

**Python chr()**

It returns character/string equivalent of a number. (internally it assumes inter as ascii value and convert them into it equivalent character).

The valid range of the integer is from 0 through 1,114,111.

print(chr(97)) #a

print(chr(11)) #♂

print(chr(57)) #9

**WAP a program to get values and key in a tupple or list of dictionary.**

Let say dictionary variable name is d

Key\_tuple=tuple(d.keys()) // get keys in tuple

Key\_list=list(d.keys() // get keys in list

*Keys()or values()method of dictionary gives dictionary view that will not be iterable. We can convert them to any iterable data using required methos(list(), tuple())*

**Create a function that converts a date formatted as MM/DD/YYYY to YYYYDDMM**

**WAP to create a dictionary using below keys and values.**

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

Method 1:

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

d={}

for i in range(len(keys)):

    d[keys[i]]=values[i]

[print(d)]

Method 2:

d=dict(zip(keys,values))

print(d)

**What is python dictionary comprehension?**

Dictionary comprehension is an elegant and concise way to create a new dictionary from an iterable in Python

Dictionary comprehension consists of an expression pair (key: value) followed by a for statement inside curly braces {}

Example:

# Dictionary Comprehension

squares = {x: x\*x for x in range(6)}

print(squares)

**Question**

**Create a new dictionary by extracting the following keys from a below dictionary**

given dictionary----

sampleDict = {

"name": "Kelly",

"age":25,

"salary": 8000,

"city": "New york"

}

Key to extract----

keys = ["name", "salary"]

**Method 1:** ------- Using dictionary comprehension

d={k:sampleDict[k] for k in keys}

print(d)

method 2 : ---------- Traditional way

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

keys = ["name", "salary"]

#{'name': 'Kelly', 'salary': 8000}

sample\_keys=list(sampleDict.keys())

d={}

for each in sample\_keys:

    if each in keys:

        d[each]=sampleDict[each]

print(d)

**How can we test if a given key is present in a dictionary?**

We can do it using membership operator (IN). For dictionary membership checks for key presence.

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

print(49 in squares) /// False, key 49 is not present in dictionary

**Delete set of keys from a dictionary**

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

keysToRemove = ["name", "salary"] # we have to remove these keys

for each in keysToRemove:

    del sampleDict[each]

print(sampleDict)

**Rename the given key in dictionary but value will be same.**

For given dictionary rename ‘city’ key to location

sampleDict = {

"name": "Kelly",

"age":25,

"salary": 8000,

"city": "New york"

}

For above given key change the key ‘city’ to location but value will be preserved.

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

sampleDict['location']=sampleDict.pop('city')

print(sampleDict)

**Get the key of a minimum value from the following dictionary**

For given dictionary find out the key whose value is minimum.

sampleDict = {

'Physics': 82,

'Math': 65,

'history': 75

}

sampleDict = {

  'Physics': 82,

  'Math': 65,

  'history': 75

}

d=sorted(sampleDict.items(),key=lambda x:x[1])

print(d[0][0])

**Sort a list which is list of list in which sorting will be performed by the value at position 3 of each sublist.**

<https://www.hackerrank.com/challenges/decorators-2-name-directory/problem>

input = [['Mike', 'Thomson', '20', 'M'], ['Robert', 'Bustle', '32', 'M'], ['Andria', 'Bustle', '30', 'F']]

output=[['Mike', 'Thomson', '20', 'M'], ['Andria', 'Bustle', '30', 'F'], ['Robert', 'Bustle', '32', 'M']]

input\_val = [['Mike', 'Thomson', '20', 'M'], ['Robert', 'Bustle', '32', 'M'], ['Andria', 'Bustle', '30', 'F']]

output=[['Mike', 'Thomson', '20', 'M'], ['Andria', 'Bustle', '30', 'F'], ['Robert', 'Bustle', '32', 'M']]

l1=sorted(input\_val,key=lambda x:x[2])

print(input\_val)

print(l1)

**Concatenate below two lists index-wise**

list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

Method 1:

result=[list1[i]+list2[i] for i in range(len(list1))]

Method 2:

list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

list3 = [i + j for i, j in zip(list1, list2)]

print(list3)

**Concatenate two lists in the following order**

list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

Output--

['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']

list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

result=[i+j for i,j in zip(list1,list2)]

print(result)

**Add blank space at end of each element if given list.**

list2 = ["Dear", "Sir"]

list2 = ["Dear", "Sir"]

list2=[each+' ' for each in list2]

print(list2)

**Given a two Python list. Iterate both lists simultaneously such that list1 should display item in original order and list2 in reverse order.**

list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

Method 1: Using for loop

for each in range(len(list2)):

    print(list1[each],' ',list2[-each-1])

Method 2: Using zip function

for x, y in zip(list1, list2[::-1]):

    print(x, y)

**Remove empty string from a list/iterable**

list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

output must be-

["Mike", "Emma", "Kelly", "Brad"]

Method 1 : Using filter function

list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

result=list(filter((lambda x: len(x)!=0),list1))

print(result)

**What is sorted function?**

The sorted() function sorts the elements of a given iterable in a specific order (either ascending or descending) and returns the sorted iterable as a list.

*sorted(iterable, key=None, reverse=False)*

key--- A Function to execute to decide the order. Default is None and optional

**Sort a tuple of tuples by 2nd item**

tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

tuple1 = tuple(sorted(list(tuple1), key=lambda x: x[1]))#tell to sort on index of 1

print(tuple1)

tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

result=sorted(tuple1,key=lambda x:x[1])

**Sort a dictionary by value of key**

X=any dictioanry

{k: v for k, v in sorted(x.items(), key=lambda item: item[0])}

{0: 0, 2: 1, 1: 2, 4: 3, 3: 4}

**Python any()**

The any() function returns True if any element of an iterable is True. If not, any() returns False.

any(iterable) -----Syntax

it is applicable for any iterable data(list, set, tuple, dict(if dict then checks for keys))

d = {}

print(any(d))#False

d = {'0': 'False'}

print(any(d)) # True , b/c ‘0’ is string not number

**Python all()**

The all() function returns True if all elements in the given iterable are true. If not, it returns False.

all(iterable)

this is same as any() but it check for True on all elements, if on dict then check of keys.

**Python delattr()**

The delattr() deletes an attribute from the object (if the object allows it).

delattr(object\_name, var\_name) -------- syntax

e.g -- let say we have a class XYZ and then want to reomve any class varaible then we can do it using this method

* object - the object from which name attribute is to be removed
* name - a string which must be the name of the attribute to be removed from the object

class Coordinate:

  x = 10

  y = -5

  z = 0

point1 = Coordinate()

delattr(Coordinate,'z') #deleted class variable z

print(point1.z) #will get error b/c it's deleted

Python dir()

The dir() method tries to return a list of valid attributes of the object.