Python Concept

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

Escape Sequences

\" , \n - new line , \b - backspace , \t - tab

print(r"\n") -output \n - r i used for normail tesxt

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

Variables

first letter - letter or \_

snake case writing

String

Sting indaxing

left to right - +v flow

right to left -ve flow

name = "Aman"

name[0] = A

name[-1] = n

String Slicing

string[start argument, stop argument , step argument]

reverse string

rev = name[::-1]

String function and method

----

len - to find the length of the string

length = len(name) = 4

capitalize() --- name.capitalize() = Aman

string.count(value, start, end) = name.count(A,0,-1) -- start , end are optiona;

string.endswith(value, start, end)

string.find(value, start, end) --start , end are optiona;

str.replace(old, new [, count])

Print() function

map function(

map(function\_name, variables)

\*args

it contating tuple and all values/list are stored in tulple

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

Zip function

Zip(list1, list2,list3) -- output tuple (value of list1, value of list2,list3)

Example

list1 = ["user1", "User2","User3"]

list2 = ["Aman","Ajay","Akash"]

result = zip(list1,list2) # - --output will be iterator object

print(list(result) - [(User1,Aman),(User2,Ajay),(User3,Akash)]

\*operator with ZIP

zip(\*result) - output - tuple with first element of result tuple will go into one tuple and second element of result tuple go into one tuple

Example

list(zip(\*result)) = [("user1","User2","User3"),("Aman","Ajay","Akash")]

l1, l2 = list(zip(\*result)) --- output == l1["user1", "User2","User3"] , l2 ["Aman","Ajay","Akash"]

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

any , all functuon

all

all will rturn true if all element are true else false

example

leven\_result = lambda l:all([True if i%2==0 else False for i in l])

print(leven\_result([1,2,3,4,5,6,7,8])) #-- output - false

leven\_result = lambda l:all([True if i%2==0 else False for i in l])

print(leven\_result([2,4,6,8])) # -- output Ture

any

will return true if any true value in present in list

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

Advance min() and max() function

return the maximun value

i f we want to find the string that has maximum length

list1 = ["Aman","Ajay","Akash"]

print(max(list1 , key = (lambda items: len(items))))

stu = {

"a ":{"s":90,"a":24},

"m":{"s":75,"a":19},

"r":{"s":76,"a":23}

}

print(max(stu, key = lambda itm:stu[itm]["s"]))

-----

Advance sorted function

sort method will not work for tuple

we have use sorted function , and it will return list as tuple as immutable so it will not modify tuple

example

list1 = ("YAman","Ajay","KAkash")

result = sorted(list1)

print(result)

same thing with set

{}

sorted(dictonary, key = func , reverse = true)

----

doc string

pritnt the value in """vxcbv""" inside the function

helps to understand the what function will do

syntex

function\_name.\_\_doc\_\_

---

pass function as argument

generator function\_name

yield keword is used to store value in generator

Example

n = 20

def gen\_Ex(value):

for i in range(1,value+1):

if i%2==0:

yield i

es = gen\_Ex(n)

for i in es:

print(i)

using genetarot comp

n = 20

def gen\_Ex(value):

return (i for i in range(1,value+1) if i%2==0 )

es = gen\_Ex(n)

for i in es:

print(i)

Object oriented programming

self.name = name

self.model = model

self.price = price

l1 = Laptop("HP","200TU",32000)

print(l1.name)

print(l1.model)

print(l1.price)

INSTANCE MeTHOD

class Laptop:

def \_\_init\_\_(self,name,model,price):

self.name = name

self.model = model

self.price = price

def full\_name(self): # instance method

return f"{self.name} {self.model}"

l1 = Laptop("HP","200TU",32000)

print(l1.name)

print(l1.model)

print(l1.price)

print(l1.full\_name())

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

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

**Python Dictionary**

A dictionary in Python is a collection of items accessed by a specific key rather than by index.

Dictionary is and unorder collections of data

**Note**: The keys in a dictionary must be hashable?

Hashing is the process of running an item through a specific kind of function. This function is called a "hash function". This hash function returns a unique output for a unique input value.

Integers, floating point numbers, strings, tuples, and frozensets are hashable. While lists, dictionaries, and sets other than frozensets are not.

**initialize a dictionary**

a = {'apple': 'fruit', 'beetroot': 'vegetable', 'cake': 'dessert'}

a['doughnut'] = 'snack'

print(a['apple']) - > output ---- fruit

The items in a dictionary can have any data type

example

user = {'Name': 'Aman', 'age':24}

user1 = dict{Name = 'Aman', age= 24}

user3 = {}

**access dictionary**

user['name'] -- user is dictionary and name are the key, we can get the value of name key by this

values can be stored in dictionary - anything list, dictionary, string.

**adding data to dictionary**

user = {}

user['name'] = "Aman"

**deleting data from dictionary**

dictionary.pop(key)

dictionary.popitem(key)

In keyword

It is used to check the key or value in dictionary

To check key in dictionary

If key in dictionary

To check value in dictionary

if values in dictionary.values()

**loops in dictionary**

to get key

for i in dictionary

print(i) - it will give key of a dictionary

to get values

for i in dictionary.values()

print(i) – it will print values of dictionary

**values method** -- it will provide all values of dictionary

**keys method** - it will give all key of dictionary

to print value from key

for i in dictionary.values()

dictionary[i] - it will give values of corresponding key

items method --- dictionary.items()

for key, vale in dictionary.items()

print key, value

update dictionary with another dictionary

dictonary1.update(dictonary2)

fromkeys

it is used to create dictonary with default value

example

d = dict.fromkeys([key1,key2,key3], value ) - we can use tuple inplace of list

dict.fromkeys("abc", value ) --- there willl be 3 key a,b,callable

we can use range also

dict.fromkeys(rane(0,11), value )

we can assing different values to diffenent key

dict.fromkeys([key1,key2,key3],[value1,value2,value3])

get method

dictonary.get(key)

if key is not in present in dictonary it will give None value

if we want to replace naon with other string we can do

print(d.get("key not in d", "value instead of none ")

deleting data from dictonary

Delete a single element

del a['one']

print(a)

>

d.clear() - it will clear the dictonary

to create copy of dictonary

d1 = d.copy()

Important to remember is that a key has to be unique in a dictionary, no duplicates are allowed. However, in case of duplicate keys rather

than giving an error, Python will take the last instance of the key to be valid and simply ignore the first key-value pair

Python Dictionary Comprehension

Dictionary comprehension is a method for transforming one dictionary into another dictionary. During this transformation, items within the original dictionary can be conditionally included in the new dictionary and each item can be transformed as needed.

A good list comprehension can make your code more expressive and thus, easier to read. The key with creating comprehensions is to not let them get so complex that your head spins when you try to decipher what they are actually doing. Keeping the idea of "easy to read" alive.

The way to do dictionary comprehension in Python is to be able to access the key objects and the value objects of a dictionary.

How can this be done?

Python has you covered! You can simply use the built-in methods for the same:

----------Debugging -------------------

import pdb

pdb.set\_trace()

l - to chek the current line

n- to execute current line

q - quit debugging

c -to execute all code

----------Working with FIle-----------------

Reading file

Open function

mode used empty or r

open('path' ,r) r is used to read the file -- by default mode is read method

read() - will read all text file

readline() will read one line

readlines() - will combine all line into one lines

to print all data as septate

a = f.readlines()

for line in a

print(a,end='')

f = open()

f.tell() - to check the cursor position

f.seek(integer) - changing the cursor position

f.name - fileName

f.closed() - to checkif file is closed or not return true or false

f.close() - to close file

slicing in readlines

readline()[:2]

write to file

mode used in write - w,a,r+

--w mode

weite method -

overrride the file, it will delete the current data from file and write new file

if file is not exict then it wiull create the new file and write the data

example

with open('path','w') as f

f.write

-- a mode or append mode

it will not delete current data from file and it wil add extra data into file

if file not exict it will create new file

-- r+ ---

we can read as well as write the file

it will not create new file if file is not exict

it will replac olde data with new data with lentgh of new data

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

working with CSV file

from csv import reader , Dictwriter

with open()

---------OS module

import OS

getcwd() - current working directory

os.path.exists

os.listdir()

os.walk()

os.makedirs() - multiple directory

shutil.rmtree() delete folder p

shutil.copytree("source location","destination location")

shutil.move(,)