ses 3 presentaion list tuple dict function

October 8, 2021

0.1 Assign lists

```
list a list is a collection of data which is ordered and changable & allow duplicate members|ea
```

```
A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutal
```

[ ]	creat a list	x = [1,3,'re']
list((values))	creat list	x = list(1,4,'sd')
x[index]=value	assign new value	x[4]=2
x[index]	get value	x[4]
x[start:end:step]	slice list or get multi value	x[:]
len(list)	get length	len(x)
list.append(value)	add value	x.append()
list.remove(value)	remove value	x.remove()
list.insert(index , value)	insert to position	x.insert(3,'a')
list.pop(index)	remove from position	x.pop(2)
list.reverse()	revrse list	x.reverse()
list.sort()	sort list alphaiticaly (all element ust be from the same class)	x.sort()
list.sort(reverse=True)	reverse sort	x.sort(reverse=True)

[ ]:
```
#assign
x = [1,3,3,6,1,23,5,6,7,2,2]
print(x)
```

[ ]:
```
# create empty list
y = [5, 8, 3.14, -90 , "hello"]
z = [5, 8, 3.14, -90 , "hello", [5, 9, [-30]]]
```

[ ]:
```
# you can access any ele using it's index
print(z[3]) # this will print the ele at index 3 -90
print(z[4])
print(z[5])
print(z[5][1])
print(z[5][2][0])
print(z[4][1])
```

[ ]:
```
# the negative indexing also works with lists
print(z[-1])
print(z[-1][2][0])
print(z[-1][-1][-1])
```

[ ]:
```
#assign new value
x[4]= 98
print(x)
```

[ ]:
```
#get value
print(x[2])
```

[ ]:
```
#get multi value
y= x[5:7]
print(x[2:4:1] , x[-4:] , x[:] , y)
```

[ ]:
```
#get length
print(len(x))
```

[ ]:
```
# add value
x.append(2)
print(x)
```

[ ]:
```
x = [1,5,-90]
y = [20,13]
# what if we want to append each ele from the list y to list x
# you can't say x.append(y) because the result would be [1,5,-90, [20,13]]
# method 1
for ele in y:
x.append(ele)
print(x)
```

[ ]:
```
#method 2
x = [1,5,-90]
y = [20,13]
x.extend(y)
print(x)
```

[ ]:
```
#remove value
x.remove(1)
print(x)
```

[ ]:
```
#add value in position
print(x.insert(5,12))
print(x)
```

[ ]:
```
#remove value from position
print(x.pop(4))
print(x)
```

[17]:
```
# Lists are mutable
s = ["hello ","world"]
s[1] = "m" # strings are imutable you can't modify their values
```

[ ]:
```
s = "hello world"
s[1] = "m" # strings are imutable you can't modify their values
```

[ ]:
```
#reverse list
print(x.reverse())
print(x)
```

[ ]:
```
#sort list
print(x.sort())
print(x)
```

[ ]:
```
#reverse list
print(x.sort(reverse=True))
print(x)
```

[ ]:
```
#make a list
y = list((1,4,'sd'))
print(y)
```

[ ]:
```
# sort only work if the values of the list are same class
print(y.sort())
```

0.1.1 List triks

[ ]:
```
# note try to visualize this code execution http://www.pythontutor.com/
 ↪visualize.ht
x = [1 , -90]
y = x
y[0] = -50
print(x)
print(y)
```

[ ]:
```
x = [1 , -90]
y = x.copy()
y[0] = -50
```

```
print(x)
print(y)
```

[ ]:
```
x = [1 , -90]
y = x[:] # sometime they called [:] copy operator
y[0] = -50
print(x)
print(y)
```

[ ]:
```

```

0.1.2 iterate over the list

[ ]:
```
l1 = [1,5, -90 , "hello"]
# method 1
for ele in l1: # ele is any variable name like x you can rename it whatever you␣
 ↪wan
print(ele)
print("*"*20)
# method 2 (using index)
for i in range(len(l1)):#4 > 0, 1, 2, 3
print(f"the ele stored at index {i} is {l1[i]}")
```

[ ]:
```
l1 = [ [1,5] ,[1,90], [20,-13] , [15,12]]
# method 1
summ = 0
for sub_list in l1: # ele is any variable name like x you can rename it␣
 ↪whatever yo
for ele in sub_list:
summ += ele
print(summ)
```

0.1.3 shopping cart exercise

[ ]:
```
products = [["milk", 3.5], ["tea", 50] , ["product" , 113]] # ?
total_price = 0
for product in products:
total_price = total_price + product[1]
total_price = total_price*1.14
print(f"you have added {len(products)} items.the total price for the added␣
 ↪items {r
```

0.1.4 uber exercise

[ ]:
```
!pip install geopy --user
```

we are developing app like uber and we got a request in the server-side who is waiting in the Ramses Railway the mobile app sent his location(lat , lon) and our target is to connect the rider with the nearest driver. we have a list of drivers for each driver we know his (lat,lon, car name). develop an algorithm to choose the nearest car for a certain location from a list of cars. Note: to get distance between two points Notes: import geopy.distance location1 = [30.063249588012695 , 31.24689292907715] location2 = [30.068357467651367 , 31.344741821289062] distance= geopy.distance.vincenty(location1, location2).km # the distance in km print(distance) # this should print 9.451816024957292

[18]:
```
import geopy.distance
location1 = [30.063249588012695 , 31.24689292907715]
location2 = [30.00708770751953 , 31.408044815063477]
distance= geopy.distance.geodesic(location1, location2).km # the distance in km
print(distance) # this should print 9.451816024957292
```

```
16.743912725207693
```

[ ]:
```
import geopy.distance
rider_location = [30.063249588012695 , 31.24689292907715] # (lat , lon) Ramses␣
↪Rail
surrounding_drivers = [
[30.068357467651367 , 31.344741821289062 , "car2"],
# City Center, Makram Eba
[30.04442596435547 , 31.235675811767578 , "car1"],
# Tahrir Square
[30.0284366607666
, 31.408044815063477 , "car3"],
# Cairo Festival City Mal
[30.00708770751953 , 31.408044815063477 , "car4"]
# Mall of Arabia, 6th of O
]
# distance= geopy.distance.geodesic(location1, location2).km # the distance in␣
↪km
min_dis_car = 9999999999999999
car_name = ""
for driver in surrounding_drivers:
driver_loc = driver[:2]
distance= geopy.distance.geodesic(rider_location, driver_loc).km
if distance < min_dis_car:
car_name = driver[2]
min_dis_car = distance
print(car_name)
```

0.2 List manuplation

```
[ ]: x = [5 for i in range(10)]
     print(x)
```

```
[ ]: L = [x**2 for x in range(5)]
     print(L)
```

```
[ ]: x = [10 for i in range(3)]
     print(x)
```

```
[ ]: # Conditionals in List Comprehension
     number_list = [ x for x in range(20) if x % 2 == 0]
     print(number_list)
```

```
[ ]: # if...else With List Comprehension
     obj = ["Even" if i%2==0 else "Odd" for i in range(10)]
     print(obj)
```

```
[ ]: number_list = []
     for x in range(20):
         if x % 2 == 0:
             number_list.append(x)
     print(number_list)
```

```
[ ]: obj = []
     for i in range(10):
         obj.append("Even" if i%2==0 else "Odd")
     print(obj)
```

```
[ ]: obj = []
     for i in range(10):
         if i%2==0 :
             obj.append("Even")
         else:
             obj.append("Odd")
     print(obj)
```

0.3 Assign tuple

tuple :	a tuple is a collection of data which is ordered and unchangable & allow duplicate members	each member has an index & the first member's index is 0
(values)	creat a tuple	x = (1,3,'re')
tuple((values))	creat tuple	x = list((1,4,'sd'))
x[index]	get value	x[4]

tuple :	a tuple is a collection of data which is ordered and unchangable & allow duplicate members	each member has an index & the first member's index is 0
len(tuple)	get length	len(x)

```
[14]: #assign
x = (1,3,3,6,1)
print(x)
```

```
(1, 3, 3, 6, 1)
```

```
[ ]: #slice tuple
y= x[5:7]
print(y)
```

```
[ ]: #make a list
y = tuple((1,4,'sd'))
print(y)
```

```
[ ]: #get value
print(x[2])
```

```
[ ]: #get length
print(len(x))
```

0.4 Assign dictionary

dictionary : a dict is a collection of data which is unordered and indexed & no duplicate members allow each value has a key

Dictionaries and lists share the following characteristics:

```
Both are mutable.
Both are dynamic. They can grow and shrink as needed.
Both can be nested. A list can contain another list. A dictionary can contain another dictiona
A dictionary can also contain a list, and vice versa.
```

Dictionaries differ from lists primarily in how elements are accessed:

```
List elements are accessed by their position in the list, via indexing.
Dictionary elements are accessed via keys.
```

{ }	creat a dict , nested dict	x = {‘in’:1,‘as’:3,‘aw’:‘re’} / x = dict([(1, ‘Geeks’), (2, ‘For’)]) / Dict = {1: ‘Geeks’, 2: ‘For’, 3:{‘A’ : ‘Welcome’, ‘B’ : ‘To’, ‘C’ : ‘Geeks’}}
dict(())	creat dict	x = dict((in=1,as=3,aw=‘re’))
dict[‘key’] / x.get(key)	get value	x[‘in’] / x.get(‘in’)
dict.items	get values	x.items
x.keys	get keys	x.keys
len(dict)	get length of keys	len(x)
dict[’key]=‘value’	add key & value	x[‘in’]= 324
dict.pop(index) / del(dict[‘key’])	remove value and key from position	x.pop(2) / del(x[‘age’])
cidt.clear()	clear dict	x.clear()
dict.update({‘new_key1’:new_value1,‘new_key2’:new_value2})	update multiple of a dict	dict.update({‘c’:3,‘d’:4})

[ ]:
```
#create dict
x = {'in':1,'as':3,'aw':'re'}
```

[ ]:
```
y = dict([(1, 'hello'), (2, 'world')])
```

[ ]:
```
Dict = {1: 'hello', 2: 'world', 3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'python'}}
```

Building a Dictionary Incrementally

[ ]:
```
person = {}
person['fname'] = 'mohamed'
person['fname'] = 'ahmed'
person['lname'] = 'yehia'
person['age'] = 30
person['preferred anime'] = ['black clover', 'hunter', 'attack on titans']
print(person)
```

[ ]:
```
person['preferred anime'] = [{'rating':7, 'name':'black clover'},
{'rating':9, 'name':'hunter'},
{'rating':9, 'name':'attack on titans'}]
```

[ ]:
```
print(person)
```

[ ]:
```
person['preferred anime'][1]["rating"]
```

[ ]:
```

```

[ ]:
```
#get value
print(x['in'])
print(x.get('in'))
```

```
[ ]: #check if the key existed in dict

     d = {'name':'Bob', 'age':25, 'job':'dev', 'city':'New York', 'email':'bob@web.
     ↪com'}
     if "name" in d:
     print(d['name'])
     if "phone-number" not in d:
     d["phone-number"] = "000000000000000000"
     print(d)
```

```
[ ]: #get all items
     print(x.items())
```

```
[ ]: #get all keys
     print(x.keys())
```

```
[ ]: #get numbers of keys
     print(len(x))
```

```
[ ]: #add value to key
     x['in']= 324
     print(x)
```

```
[ ]: #remove value and key
     print(x.pop('as') )

     del(x['in'])

     print(x)
```

```
[ ]: #clear dict
     print(x.clear())
```

```
[ ]:
```

```
[ ]: #iterate over dict
     #method 1
     d = {'name':'Bob', 'age':25, 'job':'dev', 'city':'New York', 'email':'bob@web.
     ↪com'}
     for ahmed in d: # key is just var name you can choose whatever you want
     print(f"the {ahmed} pointed to the value {d[ahmed]}")
```

```
[ ]: #method 2
     for key, value in d.items():
     print(f"the {key} pointed to the value {value}")
```

0.5 count words

```
[ ]: sample = '''A wonderful serenity has taken possession of my entire soul, like␣
     ↪ these sweet mornings of spring which I enjoy with my whole heart. I am␣
     ↪ alone, and feel the charm of existence in this spot, which was created for␣
     ↪ the bliss of souls like mine. I am so happy, my dear friend, so absorbed in␣
     ↪ the exquisite sense of mere tranquil existence, that I neglect my talents.␣
     ↪I␣
     ↪ should be incapable of drawing a single stroke at the present moment; and␣
     ↪ yet I feel that I never was a greater artist than now. When, while the␣
     ↪ lovely valley teems with vapour around me, and the meridian sun strikes the␣
     ↪ upper surface of the impenetrable foliage of my trees, and but a few stray␣
     ↪ gleams steal into the inner sanctuary, I throw myself down among the tall␣
     ↪ grass by the trickling stream; and, as I lie close to the earth, a thousand␣
     ↪ unknown plants are noticed by me: when I hear the buzz of the little world␣
     ↪ among the stalks, and grow familiar with the countless indescribable forms␣
     ↪ of the insects and flies, then I feel the presence of the Almighty, who␣
     ↪ formed us in his own image'''
```

```
[ ]: # put your code here
     s = sample#"hello world hello"
     words = s.split(" ")
     result = {}
     for word in words:
     if word not in result:
     result[word] = 1
     else:
     result[word] = result[word] + 1
     print(result)
```

1 JSON Data in Python

is JavaScript Object Notation , and is used to store and transfer the data

Read, Write and Parse JSON using Python

```
Python read JSON file
    json.load(file_object)

Python Writing JSON to a file
    json.dump(dict, file_pointer)
```

```
[ ]: import json

     # Opening JSON file
     f = open('data.json',)
```

```
# returns JSON object as
# a dictionary
data = json.load(f)

# Iterating through the json
# list
for i in data['emp_details']:
    print(i)

# Closing file
f.close()
```

[ ]:
```
# Data to be written
dictionary ={
    "name" : "sathiyajith",
    "rollno" : 56,
    "cgpa" : 8.6,
    "phonenumber" : "9976770500"
}

with open("data.json", "w") as outfile:
    json.dump(dictionary, outfile)
```

[ ]:
```
# JSON data:
x =  '{ "organization":"GeeksForGeeks", "city":"Noida", "country":"India"}'

# python object to be appended
y = {"pin":110096}

# parsing JSON string:
z = json.loads(x)

# appending the data
z.update(y)

# the result is a JSON string:
print(json.dumps(z))
```

2 PrettyPrint JSON Data

JSON file without Pretty Printing it:

```
{"id": 1, "name": "Jessa Duggar", "class": 9, "attendance": 75, "subjects": ["English", "Geome
"jessa@pynative.com"}
```

Pretty-Printed JSON data into a file with indent=0

```
{
"id": 1,
"name": "Jessa Duggar",
"class": 9,
"attendance": 75,
"subjects": [
"English",
"Geometry"
],
"email": "jessa@pynative.com"
}
```

Pretty-Printed JSON data into a file with indent=4

```
{
    "id": 1,
    "name": "Jessa Duggar",
    "class": 9,
    "attendance": 75,
    "subjects": [
        "English",
        "Geometry"
    ],
    "email": "jessa@pynative.com"
}
```

[ ]:
```python
import json

print("Writing Indented and Pretty-printed JSON formatted data into a file")

student = {
    "id": 1,
    "name": "Jessa Duggar",
    "class": 9,
    "attendance": 75,
    "subjects": ["English", "Geometry"],
    "email": "jessa@pynative.com"
}

with open("studentWithoutPrettyPrint.json", "w") as write_file:
    json.dump(student, write_file)
print("Done writing JSON data into file without Pretty Printing it")

with open("studentWithPrettyPrint2.json", "w") as write_file:
    json.dump(student, write_file, indent=0)
print("Done writing PrettyPrinted JSON data into file with indent=0")

with open("studentWithPrettyPrint3.json", "w") as write_file:
    json.dump(student, write_file, indent=4, sort_keys=True)
print("Done writing Sorted and PrettyPrinted JSON data into file")
```

[ ]:
```
import requests
import json
api_key = "204c762825b47ea8dd6859fb9e5b344b"
base_url = "http://api.openweathermap.org/data/2.5/weather?"
complete_url = base_url + "appid=" + api_key + "&q=cairo"
response = requests.get(complete_url)
received_dict = response.json()
# the above code is not important
print(received_dict)
```

3 Function

```
def function_name( inputs ):    #A function is a set of statements that take inputs, do some sp
```

pass by value

```
a value will be passed to function if it was from the following data type ( string , int , floa
```

Pass by Reference

```
a refrence of function unput will be passed to function if it was from the following data type
      ex : passig a list to fuction to modify an element

            if  a list or a dict wanted to be passed by value it can be done by (list.copy(), d
```

***args**
pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list.

****kwargs pass a keyworded, variable-length arguments list.**

yield

```
a yield statement suspends function's execution and sends a value back to the caller, but retai
state to enable function to resume where it is left off.
```

[ ]:
```
# pass by value
def myFun(x):

   x = 20

lst = 10
myFun(lst)
print(lst)
```

[ ]:
```
# Here x is a new reference to same list lst
def myFun(x):
   x[0] = 20

# Driver Code (Note that lst is modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun(lst);
print(lst)
```

[ ]:
```
#pass by refrence
def myFun_2(x):
   x = [20, 30, 40]

# Driver Code (Note that lst is modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
myFun_2(lst)
print(lst)
```

[ ]:
```
# how x is passed insid a function by refrence
def myFun_3(x):
   x[4] = 20
   print(id(x))
   x.clear()
   x.extend( [20, 30, 40,23,54,74,12] )
   print(id(x))
   x = [50,45,32]
   print(id(x))
# Driver Code (Note that lst is modified
# after function call.
lst = [10, 11, 12, 13, 14, 15]
print(id(lst))
myFun_3(lst)
print(lst)
```

[ ]:
```
# *args
def myFun_3(arg1, *argv):
    print ("First argument :", arg1)
    for arg in argv:
        print("Next argument through *argv :", arg)

myFun_3('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

[ ]:
```
#**kwargs
def myFun_4(**kwargs):
    for key, value in kwargs.items():
        print ("%s == %s" %(key, value))

# Driver code
myFun_4(first ='Geeks', mid ='for', last='Geeks')
```

[ ]:
```
#**Kwargs and *args
def myFun_5(*args,**kwargs):
    print("args: ", args)
    print("kwargs: ", kwargs)


# Now we can use both *args ,**kwargs to pass arguments to this function :
myFun_5('geeks','for','geeks',first="Geeks",mid="for",last="Geeks")
```

[ ]:
```
#Yield
def nextSquare():
    i = 1
    # An Infinite loop to generate squares
    while True:
        yield i*i
        i += 1  # Next execution resumes
                # from this point

# Driver code to test above generator
# function
for num in nextSquare():
    if num > 100:
         break
    print(num)
```

4 First Class functions in Python

Functions are objects

we are assigning function to a variable. This assignment doesn't call the function. It takes th

Functions can be passed as arguments to other functions

unctions are objects so we can pass them as arguments to other functions

Functions can return another function

functions are objects so we can return a function from another function

[9]:
```
# can be treated as objects
def shout(text):
    return text.upper()
```

```
print (shout('Hello') )

yell = shout

print (yell('Hello') )
```

```
HELLO
HELLO
################################################################################
########################

HI, I AM CREATED BY A FUNCTION PASSED AS AN ARGUMENT.
hi, i am created by a function passed as an argument.
################################################################################
########################

25
```

[ ]:
```
# can be passed as arguments to other functions
def shout_2(text):
    return text.upper()

def whisper_2(text):
    return text.lower()

def greet_2(func):
    # storing the function in a variable
    greeting = func("Hi, I am created by a function passed as an argument.")
    print (greeting)

greet_2(shout_2)
greet_2(whisper_2)
```

[ ]:
```
# Functions can return another function

def create_adder(x):
    def adder(y):
        return x+y

    return adder

add_15 = create_adder(15)

print( add_15(10) )
```

5 Global Variable

the one that are defined and declared outside a function and we need to use them inside a fun

Use of global keyword:

To access a global variable inside a function there is no need to use global keyword.

Without global keyword

output will conduct an error because we are trying to assign a value to a variable in an oute

With global keyword

[10]:
```
a = 1

# Uses global because there is no local 'a'
def f():
    print('Inside f() : ', a)

# Variable 'a' is redefined as a local
def g():
    a = 2
    print('Inside g() : ', a)

# Uses global keyword to modify global 'a'
def h():
    global a
    a = 3
    print('Inside h() : ', a)

# Global scope
print('global : ',a)
f()
print('global : ',a)
g()
print('global : ',a)
h()
print('global : ',a)
```

```
global :  1
Inside f() :  1
global :  1
Inside g() :  2
global :  1
Inside h() :  3
global :  3
```

```
################################################################################
########################

Value of x inside a function : 20
Value of x inside a function : 30
Value of x outside a function : 20
Value of x outside a function : 30
```

[ ]:
```
# value inside a function

x = 15
y = 30
def change():

    # using a global keyword
    global x

    # increment value of a by 5
    x = x + 5
    ####  y = y+10 (ucommenting this line will result an error as  y not␣
 ↪defined inside a function as global)
    print("Value of x inside a function :", x)
    print("Value of x inside a function :", y)

change()
print("Value of x outside a function :", x)
print("Value of x outside a function :", y)
```

5.1 Exercise 1

Code 1: Create a config.py file to store global variables:

```
x = 0
y = 0
z ="none"
```

Code 2: Create a modify.py file to modify global variables:

```
import config
config.x = 1
config.y = 2
config.z ="geeksforgeeks"
```

Code 3: Create a main.py file to modify global variables:

```
import config
import modify
print(config.x)
print(config.y)
print(config.z)
```

[12]:
```
# how to use if and function as a loop

def calculateSum(num):
    print("outside if" , num)
    if num:
        return num + calculateSum(num-1)
        print("insid if",num)
    else:
        return 0

res = calculateSum(10)
print(res)
```

```
outside if 10
outside if 9
outside if 8
outside if 7
outside if 6
outside if 5
outside if 4
outside if 3
outside if 2
outside if 1
outside if 0
55
```

[13]:
```
# two ways to iterate over a string

for char in sample_str: # char is gust any variable name you can use i
    print(char)

print("***************************")
for x in range(len(sample_str)):
    print(sample_str[x])
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-76db9e0fffca> in <module>
      1 # two ways to iterate over a string
      2
----> 3 for char in sample_str: # char is gust any variable name you can use i
      4     print(char)
      5

NameError: name 'sample_str' is not defined
```

[ ]:

```

```