Name : K. Praveen Kumar

Assignment-5(16-12-2023)

Nested Function:  
A function that is defined inside another function is known as the inner function or nested function. Nested functions are able to access variables of the enclosing scope.

Example:

Example:  
def talk(phrase):

    def say(word):

        print(word)

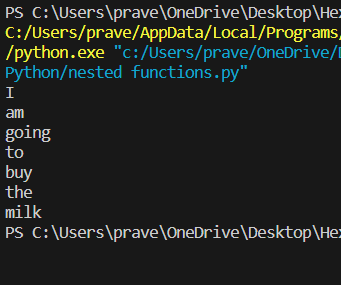
    words=phrase.split(' ')

    for word in words:

        say(word)

talk('I am going to buy the milk')

output:



**Anonymous functions:**

An anonymous function means that a function is without a name and the lambda keyword is used to create anonymous functions.

def cube(y):

    return y\*y\*y

lambda\_cube = lambda y: y\*y\*y

print("Using function defined with `def` keyword, cube:", cube(5))

print("Using lambda function, cube:", lambda\_cube(5))

output:



**Return function in Python:**

The return statement is used to exit from a function and go back to the function caller and return the specified value.

def hello(name):

    print('hello'+name+'!')

    return name,'beau',8

print(hello('syd'))

Output:



**Pass by Reference and pass by value:**

In python every variable is a reference and when we pass a variable to a function, a new reference to the object is created.

def myfun(x):

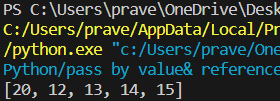
    x[0]=20

lst=[11,12,13,14,15]

myfun(lst)

print(lst)

output:



When we pass a reference and change the received reference to something else, the connection between the passed and received parameter is broken.

def f1(x):

    x=[20,30,40]

list=[1,2,3,4,5,6]

f1(list)

print(list)

output:  


**Map() Function:**

Returns a map object of the results after applying the given function to each item of a given literals(list, tuple, etc…)

Syntax: map(fun, iter)

Parameters:

fun: it is a function to which map passes each element of given iterable.

Iter: It is iterable which is to be mapped.

Example:

def add(n):

    return n+n

num=(1,2,3,4,5)

res=map(add,num)

print(list(res))

output:



Modify string using map():  
l=['sat','sun','mon','tue']

test=list(map(list,l))

print(test)

output:  


if with map():

def double\_even(n):

    if n%2==0:

        return n\*2

    else:

        return n

numbers=[11,24,45,35,78]

res=list(map(double\_even,numbers))

print(res)

**output:  
**

**String Functions in Python:**

#capitalize

sentence\_1 = "mY name is YUVRAJ"

sentence\_2 = "MY name is Ansul"

capitalized\_string = sentence\_1.capitalize()

print("Sentence 1 output -> ", capitalized\_string)

capitalized\_string = sentence\_2.capitalize()

print("Sentence 2 output -> ", capitalized\_string)

#count

message = 'GFG KARLO HO JAYEGA'

print('Number of occurrence of G:', message.count('G'))

#Find

message = 'Yuvraj is my name'

print(message.find('is'))

#lower

message = 'HEXAFORHEXA IS A COMPUTER SCIENCE PORTAL'

print(message.lower())

#upper

message = 'HexaforHexa is a computer science portal'

print(message.upper())

#Replace

text = 'subway surfer'

replaced\_text = text.replace('s', 't')

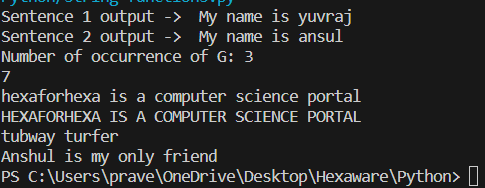
print(replaced\_text)

#join

text = ['Anshul', 'is', 'my', 'only', 'friend']

print(' '.join(text))

**output:**

****

**Modules in Python standard library:**

**1.DateTime module**

**2.Time module**

**3.Calendar module**

import time

# This is required to include time module.

ticks = time.time()

print ("Number of ticks since 12:00pm, december 16, 2023:", ticks)

print(time.localtime())

localtime = time.localtime(time.time())

print ("Local current time :", localtime)

#getting the formatted time

localtime = time.asctime( time.localtime(time.time()) )

print ("Local current time :", localtime)

import calendar

cal=calendar.month(2024,3)

print('calender:')

print(cal)

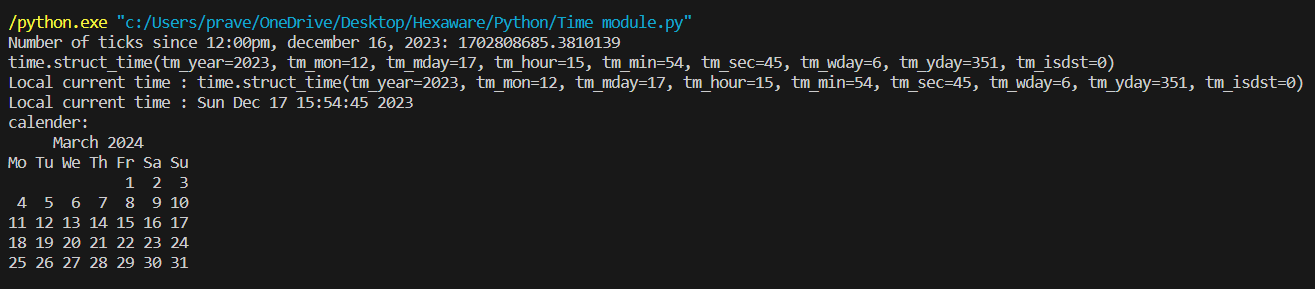
#calendar.calendar(year,w=2,l=1,c=6)

calendar.firstweekday()

calendar.isleap(2013)

calendar.leapdays(2014,2015)

**output:**

****

**Datetime module:**

from datetime import date

# Getting min date

mindate = date.min

print("Minimum Date:", mindate)

# Getting max date

maxdate = date.max

print("Maximum Date:", maxdate)

Date1 = date(2023, 4, 20)

print("Year:", Date1.year)

print("Month:", Date1.month)

print("Day:", Date1.day)

print (date.today())

d1=date.fromisoformat('2023-04-20')

print (d1)

d2=date.fromisoformat('20230420')

print (d2)

d3=date.fromisoformat('2023-W16-4')

print (d3)

d = date.fromordinal(738630) # 738630th day after 1. 1. 0001

print (d)

print (d.timetuple())

# Methods related to formatting string output

print (d.isoformat())

print (d.strftime("%d/%m/%y"))

print (d.strftime("%A %d. %B %Y"))

print (d.ctime())

print ('The {1} is {0:%d}, the {2} is {0:%B}.'.format(d, "day", "month"))

# Methods for to extracting 'components' under different calendars

t = d.timetuple()

for i in t:

   print(i)

ic = d.isocalendar()

for i in ic:

   print(i)

# A date object is immutable; all operations produce a new object

print (d.replace(month=5))

from datetime import datetime

dt = datetime.now()

print("Day: ", dt.day)

print("Month: ", dt.month)

print("Year: ", dt.year)

print("Hour: ", dt.hour)

print("Minute: ", dt.minute)

print("Second: ", dt.second)

from datetime import timedelta

delta = timedelta(

   days=100,

   seconds=27,

   microseconds=10,

   milliseconds=29000,

   minutes=5,

   hours=12,

   weeks=2

)

# Only days, seconds, and microseconds remain

print (delta)

**Output:**Minimum Date: 0001-01-01

Maximum Date: 9999-12-31

Year: 2023

Month: 4

Day: 20

2023-12-17

2023-04-20

2023-04-20

2023-04-20

2023-04-20

time.struct\_time(tm\_year=2023, tm\_mon=4, tm\_mday=20, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday=3, tm\_yday=110, tm\_isdst=-1)

2023-04-20

20/04/23

Thursday 20. April 2023

Thu Apr 20 00:00:00 2023

The day is 20, the month is April.

2023

4

20

0

0

0

3

110

-1

2023

16

4

2023-05-20

Day: 17

Month: 12

Year: 2023

Hour: 15

Minute: 57

Second: 40

114 days, 12:05:56.000010

PS C:\Users\prave\OneDrive\Desktop\Hexaware\Python>

**Lambda Functions:**

These are anonymous function means that the function is without a name.

Syntax: lambda arguments: expression

The statements will be taken in a single line only by using lambda function instead of blocks.

str1 = 'HexaforHexa'

upper = lambda string: string.upper()

print(upper(str1))

**output:**

****

**Condition check using lambda:**

format\_numeric = lambda num: f"{num:e}" if isinstance(num, int) else f"{num:,.2f}"

print("Int formatting:", format\_numeric(1000000))

print("float formatting:", format\_numeric(999999.789541235))

**output:**

****

**Difference between Lambda function and def keyword:**

The code defines a cube function using both the **‘def'** keyword and a lambda function. It calculates the cube of a given number (5 in this case) using both approaches and prints the results. The output is 125 for both the **‘def'** and lambda functions, demonstrating that they achieve the same cube calculation.

def cube(y):

    return y\*y\*y

lambda\_cube = lambda y: y\*y\*y

print("Using function defined with `def` keyword, cube:", cube(5))

print("Using lambda function, cube:", lambda\_cube(5))

**Output:**

****

**Lambda with multiple statements:**

Lambda function doesn’t allow multiple statements. However, we can create 2 lambda functions and then call the other lambda function as a parameter to the first function.

List = [[2,3,4],[1, 4, 16, 64],[3, 6, 9, 12]]

sortList = lambda x: (sorted(i) for i in x)

secondLargest = lambda x, f : [y[len(y)-2] for y in f(x)]

res = secondLargest(List, sortList)

print(res)

output:

