python™

. Moatasem Elsayed

Bio

- Embedded Linux Software Engineer

- Embedded Software Engineer

- Embedded Software Engineer

- Founder & CEO

- Mentoring For Graduation Project +40

- Instructor at Embedded Systems 75+ G

Eng.moatasem.9@gmail.com

01112932885

Content

Containers

Dictionary
Pass by value
Pass by ref
DateTime
Math
OOP
files
read
Write
Csv
Embedded
try/except/else/finally
tasks

Tasks

Write a code to suggest automatically activates for you

https://www.boredapi.com/api/activity

Get your public IP   https://api.ipify.org/?format=json

```python
from time import sleep
import requests
while True:
    url=requests.get("https://www.boredapi.com/api/activity")
    print(url.json()['activity'])
    sleep(2)
```

```python
112    ###########################################3
113    import requests
114    url=requests.get("https://api.ipify.org/?format=json")
115    print(url.text)
116
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 3> p
{"ip":"156.208.255.89"}
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 3>
```

Dictionary Method

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Cont ..

1
2
3
4
5
6
7
8
9
10
11
12
13
14

How To Study !!

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and val
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value
keys()	Returns a list containing the dictionary's keys

```python
241  car = {
242      "brand": "Ford",
243      "model": "Mustang",
244      "year": 1964
245  }
246
247  car.clear()
248
249  print(car)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded L:
{}
PS D:\Embedded System\Embedded L:
```

```python
car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 196
}

car.clear()

print(car)
```

Copy	Ctrl+C
Copy link to highlight	
Search Google for "car = { "brand": "Ford", "model": "Mustang",..."	
Print...	Ctrl+P
AdBlocker: AdBlock for Chrome	▶

Cont ..

```
40
41  car = {
42      "brand": "Ford",
43      "model": "Mustang",
44      "year": 1964
45  }
46  # car.clear()
47  print(car)
48  x = car.keys()
49  print(x)
50  x=car.values()
51  print(x)
```

```
PS D:\Embedded System\Embedded Linux\My presentation\0
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
dict_keys(['brand', 'model', 'year'])
dict_values(['Ford', 'Mustang', 1964])
PS D:\Embedded System\Embedded Linux\My presentation\0
```

Cont ..

```
254  team = dict([
255      ('Colorado', 'Rockies'),
256      ('Boston', 'Red Sox'),
257      ('Minnesota', 'Twins'),
258      ('Milwaukee', 'Brewers'),
259      ('Seattle', 'Mariners')])
260  team = dict(
261      Colorado='Rockies',
262      Boston='Red Sox',
263      Minnesota='Twins',
264      Milwaukee='Brewers',
265      Seattle='Mariners')
266  print(team)
```

```
287
288  thisdict = {
289    "brand": "Ford",
290    "model": "Mustang",
291    "year": 1964
292  }
293  for x, y in thisdict.items():
294    print(x, y)
295
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My pre
brand Ford
model Mustang
year 1964
PS D:\Embedded System\Embedded Linux\My pre

Cont ..

```
57  # #################################
58  thisdict = {
59    "brand": "Ford",
70    "model": "Mustang",
71    "year": 1964
72  }
73  #loop in keys
74  for x in thisdict:
75    print(x)
76  print("----------------------")
77  for x in thisdict.keys():
78    print(x)
79  print("----------------------")
80  #loop in values
81  for x in thisdict:
82    print(thisdict[x])
83  print("----------------------")
84  for x in thisdict.values():
85    print(x)
86  print("----------------------")
```

```
FS D:\Embedded System\
brand
model
year
----------------------
brand
model
year
----------------------
Ford
Mustang
1964
----------------------
Ford
Mustang
1964
----------------------
```

Cont ..

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the th
##########################
# this will print how many pairs is in the dict
print(len(thisdict))
##########################
# Adding item to the dict
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
thisdict["color"] = 1
print(thisdict)
```

```
Yes, 'model' is one of the keys in the thisdict dictionary

3

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 1}
```

Cont ..

```
19  thisdict["color"] = 1
20  print(thisdict)
21  thisdict["color"] = [1,2]t[str, Any]
22  thisdict["color"] = (1,2)
23  thisdict["color"] = {1,2}
24  print(thisdict)
25  thisdict[1] = "H"
26  thisdict[-1] = "E"
27  print(thisdict)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': {1, 2}}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': {1, 2}, 1: 'H', -1: 'E'}
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 5> 
```

Cont ..

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```python
############################################
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
del thisdict
print(thisdict)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentat
{'brand': 'Ford', 'year': 1964}
Traceback (most recent call last):
  File "D:\Embedded System\Embedded Linux\My pres
    print(thisdict)
NameError: name 'thisdict' is not defined
PS D:\Embedded System\Embedded Linux\My presentat
```

Cont ..

```python
39  # Nested Dictionaries
40  myfamily = {
41    "child1" : {
42      "name" : "Emil",
43      "year" : 2004
44    },
45    "child2" : {
46      "name" : "Tobias",
47      "year" : 2007
48    },
49    "child3" : {
50      "name" : "Linus",
51      "year" : 2011
52    }
53  }
54  print(myfamily)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   powershell + ∨ ⊞ 🗑 ∧ ×

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 5> python .\lab1.py
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', 'year'
': 2011}}
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 5>
```

```python
58  child1 = {
59      "name" : "Emil",
60      "year" : 2004
61  }
62  child2 = {
63      "name" : "Tobias",
64      "year" : 2007
65  }
66  child3 = {
67      "name" : "Linus",
68      "year" : 2011
69  }
70  myfamily = {
71      "child1" : child1,
72      "child2" : child2,
73      "child3" : child3
74  }
75  print(myfamily)
```

```python
}
print(myfamily)
print(myfamily["child1"]["name"])
print(myfamily["child1"])
print(myfamily.keys())
print(myfamily.get("child1"))
```

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Sessi
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobi
': 2011}}
Emil
{'name': 'Emil', 'year': 2004}
dict_keys(['child1', 'child2', 'child3'])
{'name': 'Emil', 'year': 2004}
PS D:\Embedded System\Embedded Linux\My presentation\01python\Sessi
```

Cont ..

```
80    ################################################
81  ∨ thisdict = {
82      "brand": "Ford",
83      "model": "Mustang",
84      "year": 1964
85    }
86    mydict = thisdict.copy()
87    thisdict["brand"]="changed"
88    print(mydict)
89    print(thisdict)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation\01py
PS D:\Embedded System\Embedded Linux\My presentation\01py
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'changed', 'model': 'Mustang', 'year': 1964}
PS D:\Embedded System\Embedded Linux\My presentation\01py
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

Pass by value in function

```python
def fun(x):
    x=5
    print("Inside Function ", x)
    print("Inside Function ",id(x))

x=10
print("before Function ",x)
print("before Function ",id(x))
fun(x)
print("After Function ",x)
print("After Function ",id(x))
```

```
before Function  10
before Function  2531591219792
Inside Function  5
Inside Function  2531591219632
After Function  10
After Function  2531591219792
```

Pass by ref through list

```python
def fun(x):
    x[0]=5
    print("Inside Function ", x)
    print("Inside Function ",id(x))

x=[10]
print("before Function ",x)
print("before Function ",id(x))
fun(x)
print("After Function ",x)
print("After Function ",id(x))
```

```
before Function  [10]
before Function  2514586035328
Inside Function  [5]
Inside Function  2514586035328
After Function  [5]
After Function  2514586035328
```

```python
import datetime
x = datetime.datetime.now()
print(x.year)
print(x.strftime("%A"))
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My presentat
June
PS D:\Embedded System\Embedded Linux\My presentat
2021
Tuesday
PS D:\Embedded System\Embedded Linux\My presentat
```

```python
import datetime
x = datetime.datetime.now()
print(x.strftime("%Y %B %d"))
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My preser
2021 October 26
PS D:\Embedded System\Embedded Linux\My preser
```

```python
import datetime
x = datetime.datetime.now()
print(x.strftime("%Y %B %d - %H:%M:%S"))
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation\01pythor
2021 October 26 - 15:26:39
PS D:\Embedded System\Embedded Linux\My presentation\01pythor
```

MATH

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
52  #math
53  x = min(5, 10, 25)
54  y = max(5, 10, 25)
55  print(x)
56  print(y)
57
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Lin
5
25
PS D:\Embedded System\Embedded Lin
```

```
57  ###################
58  x = abs(-7.25)
59  print(x)
60  x = pow(4, 3)
61  print(x)
62
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMIN

```
PS D:\Embedded System\Embed
7.25
64
PS D:\Embedded System\Embed
```

```
61  # print(x)
62  ###################
63  import math
64  x = math.sqrt(64)
65  print(x)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded
8.0
PS D:\Embedded System\Embedded
```

```
67  import math
68  x = math.ceil(1.4)
69  y = math.floor(1.4)
70  print(x) # returns 2
71  print(y) # returns 1
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Li
2
1
```

```
72  ###################
73  import math
74  x = math.pi
75  print(x)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded
3.141592653589793
PS D:\Embedded System\Embedded
```

math.acos()	Returns the arc cosine of a number
math.acosh()	Returns the inverse hyperbolic cosine of a n
math.asin()	Returns the arc sine of a number
math.asinh()	Returns the inverse hyperbolic sine of a num
math.atan()	Returns the arc tangent of a number in radi
math.atan2()	Returns the arc tangent of y/x in radians
math.atanh()	Returns the inverse hyperbolic tangent of a
math.ceil()	Rounds a number up to the nearest integer
math.comb()	Returns the number of ways to choose k ite and order
math.copysign()	Returns a float consisting of the value of the the second parameter
math.cos()	Returns the cosine of a number
math.cosh()	Returns the hyperbolic cosine of a number
math.degrees()	Converts an angle from radians to degrees

Class constructor ,destructor

```python
class Person:
    name = ""

    def __init__(self, name):
        print("Constructor is called")
        print(self)
        self.name = name
        print(self.name)

    def __del__(self):
        print("Destructor is called")


moatasem = Person("Moatasem")
```

```
moatasem@moatasem-Inspiron-3542:~/poky/HardD
Constructor is called
<__main__.Person object at 0x7ff20a8c4190>
Moatasem
Destructor is called
moatasem@moatasem-Inspiron-3542:~/poky/HardD
```

Class method constructor ,destructor

```python
class Person:
    name = ""
    age = 0

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def greeting(self):
        print("Hello")

    def __del__(self):
        print("Destructor is called")


moatasem = Person("Moatasem")
moatasem.greeting()
```

```
Destructor is called
moatasem@moatasem-Inspiron-3542
Constructor is called
Hello
Destructor is called
moatasem@moatasem-Inspiron-3542
```

__str__

```python
class Person:
    name = ""
    age = 0

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def greeting(self):
        print("Hello")

    def __str__(self):
        return ("description of class person ")

    def __del__(self):
        print("Destructor is called")


moatasem = Person("Moatasem")
moatasem.greeting()
print(moatasem)
```

```
class.py > Person > __str__
```

```
moatasem@moatasem-Inspiron-3542:~
Constructor is called
Hello
description of class person
Destructor is called
moatasem@moatasem-Inspiron-3542:~
as failed: /bin/sh: 1: ctags: not found
```

__doc__

```python
class MyClass:
    """
    This is a simple example class to demonstrate docstring formatting.

    Attributes:
        attribute1 (int): This is an example attribute that stores an integer value.
        attribute2 (str): This attribute stores a string value.

    Methods:
        method1(): This method performs some action and demonstrates how to document methods.

    Example:
        >>> obj = MyClass()
        >>> obj.method1()
        Action completed.

    """

    def __init__(self):
        """
        Constructor for the MyClass class.

        Initializes the attributes attribute1 and attribute2.
        """
        self.attribute1 = 0
        self.attribute2 = ""

    def method1(self):
        """
        Perform an action and print a message.

        This method demonstrates how to document methods with descriptions.
        """
        print("Action completed.")
💡
obj = MyClass()
print(obj.__doc__)
help(obj.method1)
print(obj.method1.__doc__)
```

Inheritance

```python
class animal:
    name = ""

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def eat(self):
        print("eat food")

    def __del__(self):
        print("Destructor is called")


class cat(animal):
    def __init__(self):
        print("Constructor is called")

    def sound(self):
        print("Meaouuu")

    def __del__(self):
        print("Destructor is called")


mycat = cat()
mycat.eat()
mycat.sound()
```

```
moatasem@moatasem-Inspiron-3542:
Constructor is called
eat food
Meaouuu
Destructor is called
moatasem@moatasem-Inspiron-3542:
```

Ordering of constructors

```python
class animal:
    name = ""

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def eat(self):
        print("eat food")

    def __del__(self):
        print("Destructor is called")


class cat(animal):
    def __init__(self, name):
        print("Constructor is called")
        super().__init__(name)

    def sound(self):
        print("Meaouuu")

    def __del__(self):
        print("Destructor is called")
        super().__del__()


mycat = cat("hera")
mycat.eat()
mycat.sound()
```

```python
class Data:
    def __init__(self):
        print(__class__)


class point(Data):
    def __init__(self, x, y, z):
        print(__class__)
        super().__init__()

        self.x = x
        self.y = y
        self.z = z


p2 = point(2, 3, 4)

print(p2.x, p2.y, p2.z)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   SERIAL MONITOR

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/mypresetation/01python/03_advanced$ pyth
<class '__main__.point'>
<class '__main__.Data'>
2 3 4
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/mypresetation/01python/03_advanced$ 
```

```
moatasem@moatasem-Inspi
Constructor is called
Constructor is called
eat food
Meaouuu
Destructor is called
Destructor is called
```

multi-level inheritance

```python
class soul:
    name = ""

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def heart(self):
        print("teeet")


class animal(soul):
    name = ""

    def __init__(self, name):
        print("Constructor is called")
        self.name = name

    def eat(self):
        print("eat food")

    def __del__(self):
        print("Destructor is called")


class cat(animal):
    def __init__(self, name):
        print("Constructor is called")
        animal.__init__(self, name)

    def sound(self):
        print("Meaouuu")

    def __del__(self):
        print("Destructor is called")
        super().__del__()
```

Private member

```python
class Base:
    def __init__(self):
        self.a = "Public Member"
        self.__c = "Private member"
    def privatefunc(self):
        return self.__c

# Creating a derived class
class Derived(Base):
    def __init__(self):

        #Calling constructor of
        #Base class
        Base.__init__(self)
        #print("Calling private member of base class: ")
        #print(self.__c)
# Driver code
obj1 = Base()
print(obj1.a)
print(obj1.privatefunc())
print(obj1.__c)
```

```python
class test:
    __m = 10

    def printm(self):
        print(self.__m)


t = test()
t.printm()
# print(t.__m) #Error
print(t._test__m)  # 10
```

```python
class test:
    __m = 10

    def printm(self):
        print(self.__m)

    def __pvfun(self):
        print("hello")


t = test()
# t.__pvfun()       # ERROR
t._test__pvfun()  # Works
```

static

```
1
2
3   class Student:
4       name = 'unknown'   # class attribute
5
6       def __init__(self):
7           self.age = 20   # instance attribute
8
9       @staticmethod
10      def tostring():
11          print('Student Class')
12
13
14  print(Student.tostring())
15
```

Declare a Property

The following declares the method as a property. This method must return the value of the property.

Example: @property decorator   ⧉ Copy

```python
class Student:

    def __init__(self, name):
        self.__name = name

    @property
    def name(self):
        return self.__name
```

Above, `@property` decorator applied to the `name()` method. The `name()` method returns the *private* instance attribute value `__name`. So, we can now use the `name()` method as a property to get the value of the `__name` attribute, as shown below.

Example: Access Property decorator

```python
>>> s = Student('Steve')
>>> s.name
'Steve'
```

```python
class Circle:
    def __init__(self, radius):
        self._radius = radius   # We use an underscore to indicate a "protected" attribute

    @property
    def radius(self):
        """Get the radius of the circle."""
        return self._radius

    @radius.setter
    def radius(self, value):
        """Set the radius of the circle, with validation."""
        if value >= 0:
            self._radius = value
        else:
            raise ValueError("Radius cannot be negative.")

    @radius.deleter
    def radius(self):
        """Delete the radius of the circle."""
        print("Deleting the radius...")
        del self._radius


# Create an instance of the Circle class
circle = Circle(5)

# Access the radius using the property
print(circle.radius)  # Output: 5

# Set the radius using the property
circle.radius = 7
print(circle.radius)  # Output: 7

# Try to set a negative radius (raises a ValueError)
try:
    circle.radius = -2
except ValueError as e:
    print(e)  # Output: Radius cannot be negative.
```

Please search ⟶

Operator overloading

```python
class Point:

    def __init__(self, xCoord=0, yCoord=0):
        self.xCoord = xCoord
        self.yCoord = yCoord

    # overload + operator
    def __add__(self, point_ov):
        return Point(self.xCoord + point_ov.xCoord, self.yCoord + point_ov.yCoord)

point1 = Point(2, 4)
point2 = Point(12, 8)
point3 = point1+point2

print(point3.xCoord)
```

Operator	Method
+	__add__(self, other)
-	__sub__(self, other)
*	__mul__(self, other)
/	__truediv__(self, othe
%	__mod__(self, other)
<	__lt__(self, other)
<=	__le__(self, other)
==	__eq__(self, other)
!=	__ne__(self, other)
>	__gt__(self, other)
>=	__ge__(self, other)

File Open

```
1  '''
2  "r" - Read - Default value. Opens a file for reading, error if the file does n
3  "a" - Append - Opens a file for appending, creates the file if it does not exi
4  "w" - Write - Opens a file for writing, creates the file if it does not exist
5  "x" - Create - Creates the specified file, returns an error if the file exists
6  "b" - Binary - Binary mode (e.g. images)
7  "t" - Text - Default value. Text mode
8  '''
9  f = open("file.txt",)#Equal to f = open("demofile.txt", "rt")
10 print(f.read())
```

```
D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .\lab1.py
ello World

edded Linux

d Luck
D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

```
1  Hello World
2  Embedded Linux
3  Good Luck
```

Files..(read)

```
1  f = open("D:\\myfiles\welcome.txt", "r")
2  print(f.read())
3
```

Return the 5 first characters of the file:

```
f = open("demofile.txt", "r")
print(f.read(5))
```

```
f = open("demofile.txt", "r")
for x in f:
    print(x)
```

```
10
11
12
13
14
```

```
test.py > ...
1  f = open("demofile2.txt", "r")
2  print(f.readline())
3  print(f.readline())
4  print(f.readline())
5  f.close()
6
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diplom
Now the file has more content!

MoatasemElsayed

Zein Moatasem Elsayed
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diplom

Read vs readlines

1

8
9
10
11
12
13
14

```
GENIE: CONVERSATION VIEW                          + ↦ ⟲ ⓘ ⚙
VARIABLES                                                    ⧉
Locals
  > special variables
  > f: <_io.TextIOWrapper name='demofile2.txt' mode='r' encoding='UTF-8'>
  > lst: ['Now the file has mor... content!\n', 'MoatasemElsayed\n', 'Zein Moatasem Elsayed']
    st: 'Now the file has more content!\nMoatasemElsayed\nZein Moatasem Elsayed'
  > Globals
```

```python
test.py > ...
1  f = open("demofile2.txt", "r")
2  lst = f.readlines()
3  print(lst)
4  f.close()
5  f = open("demofile2.txt", "r")
6  st = f.read()
7  print(st)
8  f.close()
9
```

```
11
12  f = open("demofile2.txt", "w")
13  f.write("Now the file has more content! ")
14  f.close()
15  #open and read the file after the appending:
16  f = open("demofile2.txt", "r")
17  print(f.read())
```

```
Now the file has more content!
PS D:\Embedded System\Embedded I
```

Create a file called "myfile.txt":

```python
f = open("myfile.txt", "x")
```

```python
import os
os.remove("demofile.txt")
```

```python
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```
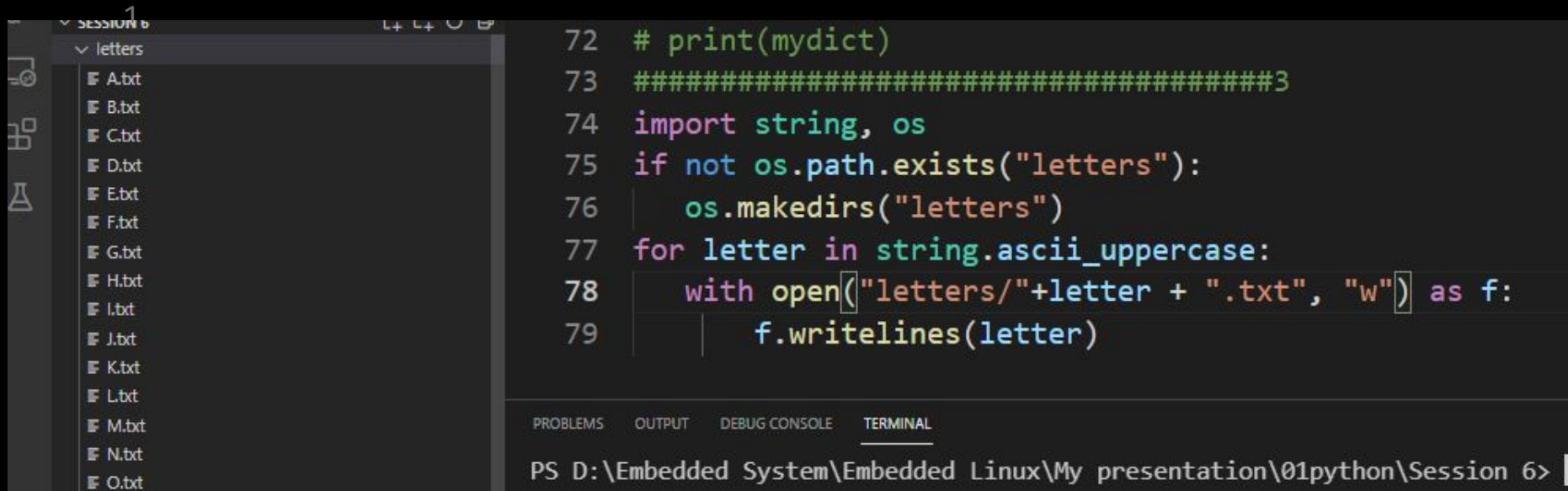
```python
import os
os.getcwd()
```

```python
import os
os.rmdir("myfolder")
```

Create list files


```
72   # print(mydict)
73   ##########################################3
74   import string, os
75   if not os.path.exists("letters"):
76       os.makedirs("letters")
77   for letter in string.ascii_uppercase:
78       with open("letters/"+letter + ".txt", "w") as f:
79           f.writelines(letter)
```

SESSION 6
letters
- A.txt
- B.txt
- C.txt
- D.txt
- E.txt
- F.txt
- G.txt
- H.txt
- I.txt
- J.txt
- K.txt
- L.txt
- M.txt
- N.txt
- O.txt

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>

1

12
13
14

```
21  ########################################
22  file = open("demofile2.txt", "r")
23  data = file.readlines()
24
25 ∨ for line in data:
26      word = line.split()
27      print(type(word))
28      print (word)
29
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
PS D:\Embedded System\Embedded Linux\My presentation\
<class 'list'>
['Now', 'the', 'file', 'has', 'more', 'content!']
PS D:\Embedded System\Embedded Linux\My presentation\
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

Trick

```
29    ###############################33
30    f = open("demofile2.txt", "r")
31    print(f.read())
32    print(f.read())
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentati
Now the file has more content!

PS D:\Embedded System\Embedded Linux\My presentati
```

with keyword

```
33    #####################################
34    with open("demofile2.txt")as f:
35        print(f.read())
36
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentati
Now the file has more content!
PS D:\Embedded System\Embedded Linux\My presentati
```

Quick task

Write a Python program to count the number of lines in a text file.

```
test.py > ...
1    f = open("demofile2.txt", "r")
2    lst = f.readlines()
3    print(len(lst))
4
```

```
36   ############################################5
37   def file_lengthy(fname):
38       with open(fname) as f:
39           for i, l in enumerate(f):
40               pass
41       return i + 1
42   print("Number of lines in the file: ",file_lengthy("file.txt"))
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .\la
Number of lines in the file:  3
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

Quick task

write a Python program to count the Number of words in a file.

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
Lest.py > ...
1  f = open("file.txt", "r")
2  st = f.read()
3  print(len(st.split()))
4
```

```
def word_count(fname):
        with open(fname) as f:
                return (f.read().split())

print("Number of words in the file :",len(word_count("file.txt")))
```

OUTPUT   DEBUG CONSOLE   TERMINAL

\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .\lab1.py
 of words in the file : 6
\Embedded System\Embedded Linux\My presentation\01python\Session 6>

Quick task

Write a Python program to write a "list" to a file.

```python
49  ####################################################3
50  color = ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
51  with open('abc.txt', "w") as myfile:
52      myfile.write(" ".join(color))
53
54  content = open('abc.txt')
55  print(content.read())
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .
Red Green White Black Pink Yellow
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

CSV

```
64  ###############################################3
65  import csv
66  reader = csv.reader(open('simplecsv.csv', 'r'))
67  mydict = {}
68  for line in reader:
69      mydict[line[0]]={'age':line[1],'color':line[2]}
70
71  print(mydict)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> py
{'ali': {'age': '12', 'color': 'blue'}, 'sayed': {'age': '99', 'color': 'gr
8', 'color': 'black'}}
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

More Generic

```python
# pip install openpyxl
import openpyxl

# Example to read data from an Excel workbook


def read_excel_file(file_path):
    workbook = openpyxl.load_workbook(file_path)
    sheet = workbook.active

    for row in sheet.iter_rows(values_only=True):
        print(row)

    workbook.close()

# Example to write data to an Excel workbook


def write_excel_file(file_path):
    workbook = openpyxl.Workbook()
    sheet = workbook.active

    data = [
        ["Name", "Age", "City"],
        ["John", 30, "New York"],
        ["Alice", 25, "London"],
        ["Bob", 35, "Paris"]
    ]

    for row in data:
        sheet.append(row)

    workbook.save(file_path)
    workbook.close()


# Example usage
write_excel_file("example.xlsx")
read_excel_file("example.xlsx")
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

example.xlsx

	A	B	C
1	Name	Age	City
2	John	30	New York
3	Alice	25	London
4	Bob	35	Paris

Create Sheet

```python
workbook_2.py > ...
1    import openpyxl
2
3    # Create a new workbook
4    workbook = openpyxl.Workbook()
5
6    # Create a new sheet
7    new_sheet = workbook.create_sheet(title="NewSheet")
8
9    # Add data to the new sheet
10   new_sheet["A1"] = "Fruit"
11   new_sheet["B1"] = "Quantity"
12   new_sheet["A2"] = "Apple"
13   new_sheet["B2"] = 10
14   new_sheet["A3"] = "Banana"
15   new_sheet["B3"] = 15
16
17   # Save the workbook
18   workbook.save("example2.xlsx")
```

Try /except


```
81
82  try:
83    print(x)
84  except:
85    print("An exception occurred")
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   **TERMINAL**

PS D:\Embedded System\Embedded Linux\My presentat
An exception occurred
PS D:\Embedded System\Embedded Linux\My presentat

try
{ Run this code }

except
{ Run this code if an
exception occurs }

else
{ Run this code if no
exception occurs }

finally
{ Always run this code }

Try /except/else/finally

```python
86
87  try :
88      print(1/0)
89  except:
90      print("Arthmatic operation Error")
91  else:
92      print("Good Software")
93  finally:
94      print("Thank You For using our app")
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation\01pytho
Arthmatic operation Error
Thank You For using our app
PS D:\Embedded System\Embedded Linux\My presentation\01pytho
```

Try /except/else/finally

```python
85  #    print("An exception occurred")
86
87  try :
88      print(1/1)
89  except:
90      print("Arthmatic operation Error")
91  else:
92      print("Good Software")
93  finally:
94      print("Thank You For using our app")
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .\lab1.py
1.0
Good Software
Thank You For using our app
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

```python
#########################################
try:
    print(1/0)
except NameError:
    print("Variable x is not defined")
except ArithmeticError:
    print("ArithmeticError Happen")
except :
    print("Something else went wrong")
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE

PS D:\Embedded System\Emb
ArithmeticError Happen
PS D:\Embedded System\Emb

```python
try:
    print(x)
except NameError:
    print("Variable x is not defined")
except ArithmeticError:
    print("ArithmeticError Happen")
except :
    print("Something else went wrong")
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation
Variable x is not defined
PS D:\Embedded System\Embedded Linux\My presentation

```python
#########################################
try:
    import aksdmngjvnqn
    pass
except NameError:
    print("Variable x is not defined")
except ArithmeticError:
    print("ArithmeticError Happen")
except :
    print("Something else went wrong")
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\Embedded System\Embedded Linux\My presentation\
Something else went wrong
PS D:\Embedded System\Embedded Linux\My presentation\

10
11
12
13
14

Question how do I know Types ?

1
2
3
4
5
6
7
8
9
10
11
12
13
14

Table of Contents

6. Built-in Exceptions
   - 6.1. Exception hierarchy

Previous topic

5. Built-in Types

Next topic

7. String Services

This Page

Show Source

Quick search

[   ] [Go]

6.1. Exception hierarchy

The class hierarchy for built-in exceptions is:

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StandardError
      |    +-- BufferError
      |    +-- ArithmeticError
      |    |    +-- FloatingPointError
      |    |    +-- OverflowError
      |    |    +-- ZeroDivisionError
      |    +-- AssertionError
      |    +-- AttributeError
      |    +-- EnvironmentError
      |    |    +-- IOError
      |    |    +-- OSError
      |    |         +-- WindowsError (Windows)
      |    |         +-- VMSError (VMS)
      |    +-- EOFError
      |    +-- ImportError
      |    +-- LookupError
      |    |    +-- IndexError
      |    |    +-- KeyError
      |    +-- MemoryError
      |    +-- NameError
      |    |    +-- UnboundLocalError
      |    +-- ReferenceError
      |    +-- RuntimeError
      |    |    +-- NotImplementedError
      |    +-- SyntaxError
      |    |    +-- IndentationError
      |    |         +-- TabError
```

https://docs.python.org/2/library/exceptions.html#exception-hierarchy

Threading

A thread is an entity within a process that can be scheduled for execution. Also, it is the smallest unit of processing that can be performed in an OS (Operating System).

Process

Thread #1   Thread #2

Time

Multithreading in Python
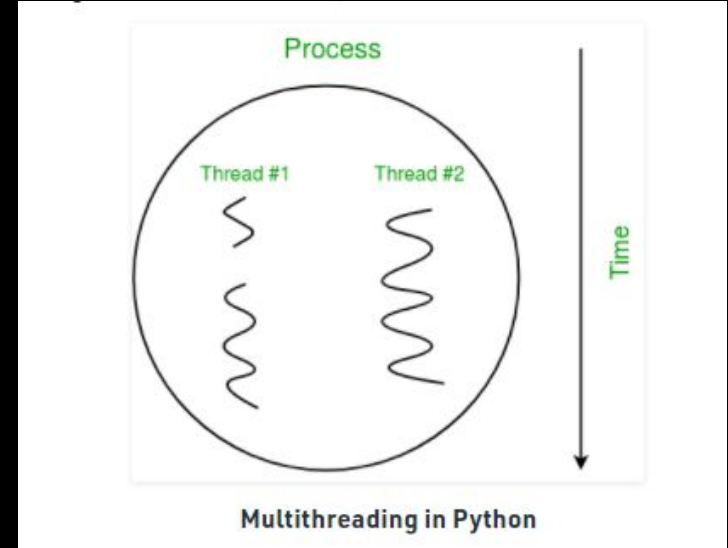
```python
3    # importing the threading module
4    import threading
5
6
7    def task1(num):
8        for i in range(0, num):
9            print("task1")
10
11
12   def task2(num):
13       for i in range(0, num):
14           print("task2")
15
16
17   if __name__ == "__main__":
18       # creating thread
19       t1 = threading.Thread(target=task1, args=(5,))
20       t2 = threading.Thread(target=task2, args=(5,))
21       # starting thread 1
22       t1.start()
23       # starting thread 2
24       t2.start()
25       # wait until thread 1 is completely executed
26       t1.join()
27       # wait until thread 2 is completely executed
28       t2.join()
29
30       # both threads completely executed
31       print("Done!")
32
```

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> python .\multitasks.p
task1
task1
task2
task2
task2
task2
task2
task1
task2
task1
task1
Done!
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```
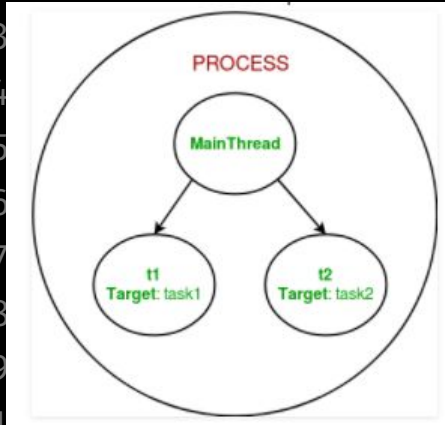
```
PS D:\Embedded System\Embedded Linux\My pres
ID of process running main program: 13984
Main thread name: MainThread
Task 1 assigned to thread: t1
ID of process running task 1: 13984
Task 2 assigned to thread: t2
ID of process running task 2: 13984
```

```python
def task1():
    print("Task 1 assigned to thread: {}".format(
        threading.current_thread().name))
    print("ID of process running task 1: {}".format(os.getpid()))


def task2():
    print("Task 2 assigned to thread: {}".format(
        threading.current_thread().name))
    print("ID of process running task 2: {}".format(os.getpid()))


if __name__ == "__main__":

    # print ID of current process
    print("ID of process running main program: {}".format(os.getpid()))

    # print name of main thread
    print("Main thread name: {}".format(threading.current_thread().name))

    # creating threads
    t1 = threading.Thread(target=task1, name='t1')
    t2 = threading.Thread(target=task2, name='t2')
```

Quick info

Threading

While both C++ and Python have threading built into the language, the results can be markedly different, depending on the problem you're solving. Frequently, threading is used to address performance problems. In C++, threading can provide a general speed-up for both computationally bound and I/O bound problems, as threads can take full advantage of the cores on a multiprocessor system.

Python, on the other hand, has made a design trade-off to use the **Global Interpreter Lock**, or the GIL, to simplify its threading implementation. There are many benefits to the GIL, but the drawback is that only one thread will be running at a single time, even if there are multiple cores.

For fun

```
76  ###########################################
77  import camelcase
78  c = camelcase.CamelCase()
79  txt = "lorem ipsum dolor sit amet"
80  print(c.hump(txt))
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation
Lorem Ipsum Dolor Sit Amet
PS D:\Embedded System\Embedded Linux\My presentation
```

12
13
14

```
81
82  text= "lorem ipsum dolor sit amet"
83  ls=list(text)
84  ls[0]=ls[0].capitalize()
85  for i in range(0,len(ls)):
86      if ls[i].islower() and ls[i-1]==" ":
87          ls[i]=str(ls[i]).capitalize()
88  #convert from list to string
89  text="".join(ls)
90  print(text)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\$
Lorem Ipsum Dolor Sit Amet
PS D:\Embedded System\Embedded Linux\My presentation\01python\$
```

Task

1

2 **Write python code**
3 **to generate Init**
4 **function of GPIO**
5 **for AVR**

6

7

8

9

10

11

12

13

14

```c
C init.c > ❽ Init_PORTA_DIR(void)
1 ∨ void Init_PORTA_DIR (void)
2    {
3        DDRA = 0b01001010;
4    }
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   **TERMINAL**
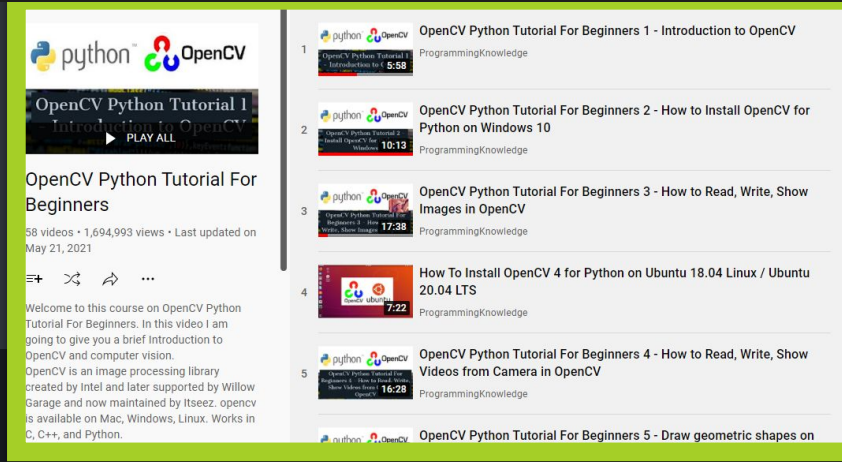
PS D:\Embedded System\Embedded Linux\My presentation\01p
Please enter Bit 0 mode: in
Please enter Bit 1 mode: out
Please enter Bit 2 mode: in
Please enter Bit 3 mode: in
Please enter Bit 4 mode: out
Please enter Bit 5 mode: in
Please enter Bit 6 mode: out
Please enter Bit 7 mode: in
PS D:\Embedded System\Embedded Linux\My presentation\01p

Task

1. **Using "Pyautogui" to open Emails and change Emails from unread to read**

2. **Run this code (% Battery and make Notification)**

```
from pynotifier import Notification
import psutil
battery = psutil.sensors_battery()
percent = battery.percent
print(percent)
Notification("Battery Percentage", str(percent)+ "%Percent Remaining", duration=10).send()
```

Task

1
2
3
4
5
6
7
8
9
10
11
12
13
14

- Write a Python program to get the command-line arguments
  Hint: import sys

Task

1
2
3 ● Python program to get the ASCII value of a
4   character.
5
6
7
8
9
10
11
12
13
14

ASCII Table

```
1  print()
2  print(ord('a'))
3  print(ord('A'))
4  print(ord('1'))
5  print(ord('@'))
6  print()
```

Task

1
2
3
4
- Python program to parse header file and read
5
   all prototypes of function and insert it into
6
   excel sheet with unique ID start with IDX0
7
8
   Example sheet:
9
10

void init();	IDX001	

11
12
13
14

Tasks (long time)

1
2
- Write a python code that manage a database for employees. Each
3   employee has a unique ID and has the following data:
4
- Name, Job and Salary. The system shall allow:
5
- 1-Add new employee
6
- 2-Print employee data
7
- 3-remove employee from the system
8
9
10 ————————————————————————————————————
11 - Write script to send an email for your friend when your device is on with its location
12
13
14

Start on hackerRank

1
2
3
4
5
6
7
8
9
10
11
12
13
14

Say "Hello, World!" With Python

Easy, Max Score: 5, Success Rate: 97.41%

Get started with Python by printing to stdout.

★ **Solve Challenge**

Python If-Else

Easy, Python (Basic), Max Score: 10, Success Rate: 91.07%

★ Solve Challenge

Arithmetic Operators

Easy, Python (Basic), Max Score: 10, Success Rate: 98.15%

★ Solve Challenge

Python: Division

Easy, Python (Basic), Max Score: 10, Success Rate: 98.83%

★ Solve Challenge

Loops

Easy, Python (Basic), Max Score: 10, Success Rate: 98.46%

★ Solve Challenge

STATUS

☐ Solved
☐ Unsolved

SKILLS

☐ Problem Solving (Basic)
☐ Python (Basic)
☐ Problem Solving (Advanced)
☐ Python (Intermediate)

DIFFICULTY

☐ Easy
☐ Medium
☐ Hard

SUBDOMAINS

☐ Introduction
☐ Basic Data Types
☐ Strings

Smart scripts

```python
45 v def Respond(voice_data):
46 v     if 'الاسم' in voice_data or 'اسم' in voice_data :
47         Bixby_Speak('الزعيم معتصم وصل يا رجاله')
48         # Bixby_Speak('Moatasem Big Boss')
49 v     if 'الوقت' in voice_data or'الساعه' in voice_data  :
50         Bixby_Speak(ctime())
51 v     if 'بحث' in voice_data or 'البحث' in voice_data :
52         search = record('العبد الله كفاية بس جوجل حكايه')
53         # search = record('what dow want to search')
54         url = 'https://google.com/search?q=' + search
55         webbrowser.get().open(url)
56         # Bixby_Speak('Here is what i Found For' + search)
57         Bixby_Speak('خلصانه يشياكه' + search)
```