



• 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

Containers

Content

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

Tasks

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

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

Cont ..

https://www.w3schools.com/python/python_ref_dictionary.asp

How To Study !!

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

```
240
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:
```

```
car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}

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",
60     "model": "Mustang",
61     "year": 1964
62 }
63 #loop in keys
64 ~ for x in thisdict:
65     print(x)
66     print("-----")
67 ~ for x in thisdict.keys():
68     print(x)
69     print("-----")
70 #loop in values
71 ~ for x in thisdict:
72     print(thisdict[x])
73     print("-----")
74 ~ for x in thisdict.values():
75     print(x)
76     print("-----")
```

```
PS D:\Embedded Systems\
brand
model
year
-----
brand
model
year
-----
Ford
Mustang
1964
-----
Ford
Mustang
1964
-----
```

Cont ..

lab1.py > ...

```
1  thisdict = {
2      "brand": "Ford",
3      "model": "Mustang",
4      "year": 1964
5  }
6  ~ if "model" in thisdict:
7      print("Yes, 'model' is one of the keys in the tl
8      #####
9      # this will print how many pairs is in the dict
10     print(len(thisdict))
11     #####
12     # Adding item to the dict
13  ~ thisdict = {
14      "brand": "Ford",
15      "model": "Mustang",
16      "year": 1964
17  }
18  thisdict["color"] = "red"
19  thisdict["color"] = 1
20  print(thisdict)
```

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

}

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

Cont ..

```
19 thisdict["color"] = 1
20 print(thisdict)
21 thisdict["color"] = [1,2]
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 ..

```
28 #####
29 thisdict = {
30     "brand": "Ford",
31     "model": "Mustang",
32     "year": 1964
33 }
34 del thisdict["model"]
35 print(thisdict)
36 del thisdict
37 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 ..

```
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 Svstem\Embedded Linux\Mv presentation\01pvthon\Session 5> █
```

Cont ..

```
57
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)
```

```

5 print(myfamily)
5 print(myfamily["child1"]["name"])
7 print(myfamily["child1"])
8 print(myfamily.keys())
9 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
```

Pass by value in function

```
1  
2  
3 1 ~ def fun(x):  
4     x=5  
5     print("Inside Function ", x)  
6     print("Inside Function ",id(x))  
7  
8 x=10  
9 print("before Function ",x)  
10 print("before Function ",id(x))  
11 fun(x)  
12 print("After Function ",x)  
13 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

lab2.py > run

```
1 def fun(x):  
2     x[0]=5  
3     print("Inside Function ", x)  
4     print("Inside Function ",id(x))  
5  
6 x=[10]  
7 print("before Function ",x)  
8 print("before Function ",id(x))  
9 fun(x)  
10 print("After Function ",x)  
11 print("After Function ",id(x))
```

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

datetime

```
39 import datetime
40 x = datetime.datetime.now()
41 print(x.year)
42 print(x.strftime("%A"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

```
43 #####
44 import datetime
45 x = datetime.datetime.now()
46 print(x.strftime("%Y %B %d"))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

```
47 #####
48 import datetime
49 x = datetime.datetime.now()
50 print(x.strftime("%Y %B %d - %H:%M:%S"))
51
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded Linux\My presentation\01python3>
2021 October 26 - 15:26:39
PS D:\Embedded System\Embedded Linux\My presentation\01python3>
```

math

```
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
D:\Embedded System\Embedded

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS D:\Embedded System\Embe
7.25
64
PS D:\Embedded System\Emk

```
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\Embed

```
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

<code>math.acos()</code>	Returns the arc cosine of a number
<code>math.acosh()</code>	Returns the inverse hyperbolic cosine of a number
<code>math.asin()</code>	Returns the arc sine of a number
<code>math.asinh()</code>	Returns the inverse hyperbolic sine of a number
<code>math.atan()</code>	Returns the arc tangent of a number in radians
<code>math.atan2()</code>	Returns the arc tangent of y/x in radians
<code>math.atanh()</code>	Returns the inverse hyperbolic tangent of a number
<code>math.ceil()</code>	Rounds a number up to the nearest integer
<code>math.comb()</code>	Returns the number of ways to choose k items from n items without regard to order
<code>math.copysign()</code>	Returns a float consisting of the value of the first parameter and the sign of the second parameter
<code>math.cos()</code>	Returns the cosine of a number
<code>math.cosh()</code>	Returns the hyperbolic cosine of a number
<code>math.degrees()</code>	Converts an angle from radians to degrees

Class constructor ,destructor

```
1 class Person:
2     name = ""
3
4     def __init__(self, name):
5         print("Constructor is called")
6         print(self)
7         self.name = name
8         print(self.name)
9
10    def __del__(self):
11        print("Destructor is called")
12
13
14    moatasem = Person("Moatasem")
15
```

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

Class method constructor ,destructor

```
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

```

class.py > Person > __str__
1 class Person:
2     name = ""
3     age = 0
4
5     def __init__(self, name):
6         print("Constructor is called")
7         self.name = name
8
9     def greeting(self):
10        print("Hello")
11
12    def __str__(self):
13        return ("description of class person ")
14
15    def __del__(self):
16        print("Destructor is called")
17
18
19 moatasem = Person("Moatasem")
20 moatasem.greeting()
21 print(moatasem)
22

```

```
moatasem@moatasem-Inspiron-3542:~$
Constructor is called
Hello
description of class person
Destructor is called
moatasem@moatasem-Inspiron-3542:~$
```

__doc__

```
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

```
class.py > ...
1  class animal:
2      name = ""
3
4      def __init__(self, name):
5          print("Constructor is called")
6          self.name = name
7
8      def eat(self):
9          print("eat food")
10
11     def __del__(self):
12         print("Destructor is called")
13
14
15     class cat(animal):
16         def __init__(self):
17             print("Constructor is called")
18
19         def sound(self):
20             print("Meaouuu")
21
22         def __del__(self):
23             print("Destructor is called")
24
25
26     mycat = cat()
27     mycat.eat()
28     mycat.sound()
29
```

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


Ordering of constructors

```
1 class animal:
2     name = ""
3
4     def __init__(self, name):
5         print("Constructor is called")
6         self.name = name
7
8     def eat(self):
9         print("eat food")
10
11    def __del__(self):
12        print("Destructor is called")
13
14
15    class cat(animal):
16        def __init__(self, name):
17            print("Constructor is called")
18            super().__init__(name)
19
20        def sound(self):
21            print("Meaouuu")
22
23        def __del__(self):
24            print("Destructor is called")
25            super().__del__()
26
27
28    mycat = cat("hera")
29    mycat.eat()
30    mycat.sound()
```

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

```
class.py > ...
1
2 class Data:
3     def __init__(self):
4         print(__class__)
5
6
7 class point(Data):
8     def __init__(self, x, y, z):
9         print(__class__)
10        super().__init__()
11
12        self.x = x
13        self.y = y
14        self.z = z
15
16
17 p2 = point(2, 3, 4)
18
19 print(p2.x, p2.y, p2.z)
20
```

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$
```

multi-level inheritance

```
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

```
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)
```

```
class test:
    __m = 10

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

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

```
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

```
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

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

Please search

```
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.
```

Operator overloading

```
1
2 class Point:
3
4     def __init__(self, xCoord=0, yCoord=0):
5         self.xCoord = xCoord
6         self.yCoord = yCoord
7
8     # overload + operator
9     def __add__(self, point_ov):
10         return Point(self.xCoord + point_ov.xCoord, self.yCoord + point_ov.yCoord)
11
12
13 point1 = Point(2, 4)
14 point2 = Point(12, 8)
15 point3 = point1+point2
16
17 print(point3.xCoord)
18
```

Operator

Method

+

`__add__(self, other)`

-

`__sub__(self, other)`

*

`__mul__(self, other)`

/

`__truediv__(self, other)`

%

`__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 not exist
3 "a" - Append - Opens a file for appending, creates the file if it does not exist
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())
```

EMS OUTPUT DEBUG CONSOLE TERMINAL

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

Hello World

Embedded Linux

Good Luck

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

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


Files..(read)

1
2
3

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

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!
```

```
MoatsemElsayed
```

```
Zein Moatsem Elsayed
```

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diplom
```

Read vs readlines

1

```
> 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
```

```
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
```

8

9

10

11

12

13

14

Files..(write)

```
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!

D:\Embedded System\Embedded

Create a file called "myfile.txt":

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

```
import os
```

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

```
import os
```

```
os.getcwd()
```

```
import os
```

```
if os.path.exists("demofile.txt"):
```

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

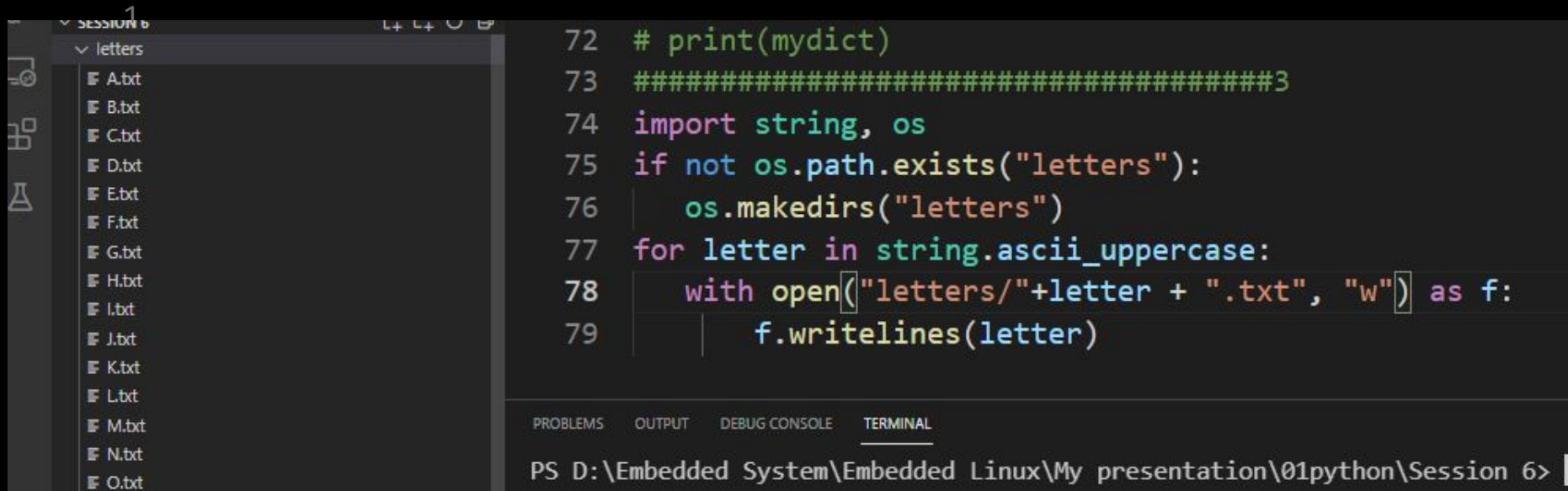
```
else:
```

```
    print("The file does not exist")
```

```
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)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

12

13

14

Split file

```
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\6
<class 'list'>
['Now', 'the', 'file', 'has', 'more', 'content!']
PS D:\Embedded System\Embedded Linux\My presentation\6
```

Trick

```
1      29 #####33
2
3      30 f = open("demofile2.txt", "r")
4
5      31 print(f.read())
6
7      32 print(f.read())
8
```

9 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

12
13
14 PS D:\Embedded System\Embedded Linux\My presentati

with keyword

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
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 presentation  
Now the file has more content!  
PS D:\Embedded System\Embedded Linux\My presentation
```


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 #####
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.

```
test.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
Number 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.

```
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

```
1      64 #####3
2      65 import csv
3      66 reader = csv.reader(open('simplecsv.csv', 'r'))
4      67 mydict = {}
5      68 for line in reader:
6      69     mydict[line[0]]={'age':line[1],'color':line[2]}
7      70
8      71 print(mydict)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
11 PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> py
12 {'ali': {'age': '12', 'color': 'blue'}, 'said': {'age': '99', 'color': 'gr
13 8', 'color': 'black'}}
14 PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6> █
```

More Generic

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

```
test.py > ...
1 # pip install openpyxl
2 import openpyxl
3
4 # Example to read data from an Excel workbook
5
6
7 def read_excel_file(file_path):
8     workbook = openpyxl.load_workbook(file_path)
9     sheet = workbook.active
10
11     for row in sheet.iter_rows(values_only=True):
12         print(row)
13
14     workbook.close()
15
16 # Example to write data to an Excel workbook
17
18
19 def write_excel_file(file_path):
20     workbook = openpyxl.Workbook()
21     sheet = workbook.active
22
23     data = [
24         ["Name", "Age", "City"],
25         ["John", 30, "New York"],
26         ["Alice", 25, "London"],
27         ["Bob", 35, "Paris"]
28     ]
29
30     for row in data:
31         sheet.append(row)
32
33     workbook.save(file_path)
34     workbook.close()
35
36
37 # Example usage
38 write_excel_file("example.xlsx")
39 read_excel_file("example.xlsx")
40
```

example.xlsx

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

Create Sheet

```
1  workbook_2.py > ...
2  1  import openpyxl
3  2
4  3  # Create a new workbook
5  4  workbook = openpyxl.Workbook()
6  5
7  6  # Create a new sheet
8  7  new_sheet = workbook.create_sheet(title="NewSheet")
9  8
10 9  # Add data to the new sheet
11 10 new_sheet["A1"] = "Fruit"
12 11 new_sheet["B1"] = "Quantity"
13 12 new_sheet["A2"] = "Apple"
14 13 new_sheet["B2"] = 10
15 14 new_sheet["A3"] = "Banana"
16 15 new_sheet["B3"] = 15
17 16
18 17 # Save the workbook
19 18 workbook.save("example2.xlsx")
```

Try /except

```
1
1 81
1 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
```

14

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

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

Try /except/else/finally

```
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> █
```



```

95 #####
96 try:
97     print(1/0)
98 except NameError:
99     print("Variable x is not defined")
100 except ArithmeticError:
101     print("ArithmeticError Happen")
102 except :
103     print("Something else went wrong")

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE

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

10
11
12
13
14

```

96 try:
97     print(x)
98 except NameError:
99     print("Variable x is not defined")
100 except ArithmeticError:
101     print("ArithmeticError Happen")
102 except :
103     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

```

95 #####
96 try:
97     import aksdmngjvnqn
98     pass
99 except NameError:
100     print("Variable x is not defined")
101 except ArithmeticError:
102     print("ArithmeticError Happen")
103 except :
104     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\

Question how do I know Types ?

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

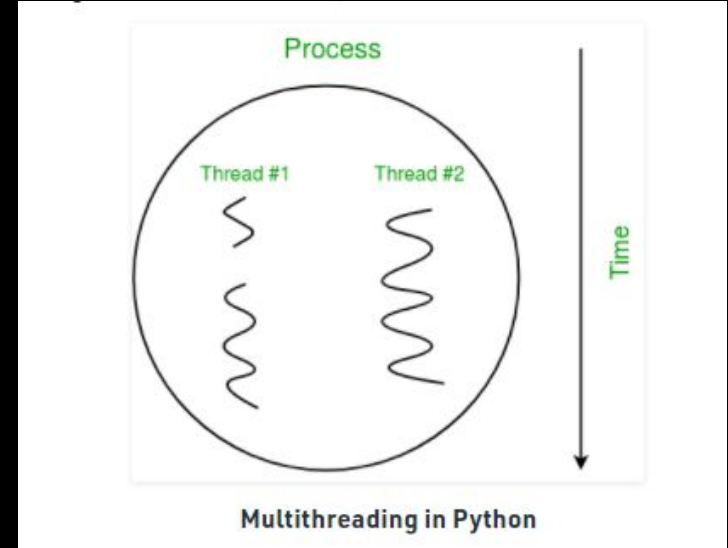
6.1. Exception hierarchy

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
            +-- SyntaxError
```

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).

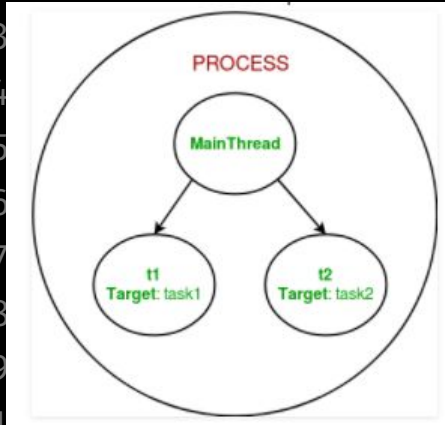


Threading (Lab)

```
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.py
task1
task1
task2
task2
task2
task2
task1
task2
task1
task1
Done!
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 6>
```

Second lab



```
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
```

```
8
9 def task1():
0     print("Task 1 assigned to thread: {}".format(
1         threading.current_thread().name))
2     print("ID of process running task 1: {}".format(os.getpid()))
3
4
5 def task2():
6     print("Task 2 assigned to thread: {}".format(
7         threading.current_thread().name))
8     print("ID of process running task 2: {}".format(os.getpid()))
9
0
1 if __name__ == "__main__":
2
3     # print ID of current process
4     print("ID of process running main program: {}".format(os.getpid()))
5
6     # print name of main thread
7     print("Main thread name: {}".format(threading.current_thread().name))
8
9     # creating threads
0     t1 = threading.Thread(target=task1, name='t1')
1     t2 = threading.Thread(target=task2, name='t2')
```

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


Task

Write python code
to generate Init
function of GPIO
for AVR

init.c > Init_PORTA_DIR(void)

```
1 void Init_PORTA_DIR (void)
2 {
3     DDRA = 0b01001010;
4 }
```

PROBLEMS

1

OUTPUT

DEBUG CONSOLE

TERMINAL

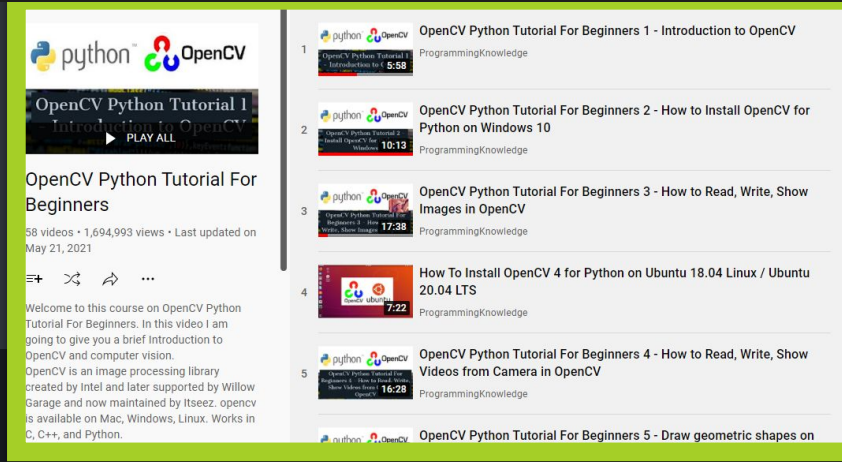
```
PS D:\Embedded System\Embedded Linux\My presentation\01p
Please enter Bit 0 mode: in
Please enter Bit 1 mode: out
Please enter Bit 2 mode: in
Please enter Bit 3 mode: in
Please enter Bit 4 mode: out
Please enter Bit 5 mode: in
Please enter Bit 6 mode: out
Please enter Bit 7 mode: in
PS D:\Embedded 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()
```



python OpenCV

OpenCV Python Tutorial 1 - Introduction to OpenCV

PLAY ALL

OpenCV Python Tutorial For Beginners

58 videos • 1,694,993 views • Last updated on May 21, 2021

Welcome to this course on OpenCV Python Tutorial For Beginners. In this video I am going to give you a brief introduction to OpenCV and computer vision.

OpenCV is an image processing library created by Intel and later supported by Willow Garage and now maintained by Itseez. opencv is available on Mac, Windows, Linux. Works in C, C++, and Python.

- 1 OpenCV Python Tutorial For Beginners 1 - Introduction to OpenCV ProgrammingKnowledge
- 2 OpenCV Python Tutorial For Beginners 2 - How to Install OpenCV for Python on Windows 10 ProgrammingKnowledge
- 3 OpenCV Python Tutorial For Beginners 3 - How to Read, Write, Show Images in OpenCV ProgrammingKnowledge
- 4 How To Install OpenCV 4 for Python on Ubuntu 18.04 Linux / Ubuntu 20.04 LTS ProgrammingKnowledge
- 5 OpenCV Python Tutorial For Beginners 4 - How to Read, Write, Show Videos from Camera in OpenCV ProgrammingKnowledge
- OpenCV Python Tutorial For Beginners 5 - Draw geometric shapes on

Task

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

```
1 prashanta@server:~$ python test.py arg1 arg2 arg3
```

Sample Output:

```
This is the name/path of the script: test.py  
( 'Number of arguments:', 4)  
( 'Argument List:', "['test.py', 'arg1', 'arg2', 'arg3']")
```

Task

- Python program to get the ASCII value of a character.

ASCII Table															
Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137		127	7F	177	

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

Task

- Python program to parse header file and read all prototypes of function and insert it into excel sheet with unique ID start with IDX0

Example sheet:

void init();	IDX001	

Tasks (long time)

- Write a python code that manage a database for employees. Each employee has a unique ID and has the following data:
 - Name, Job and Salary. The system shall allow:
 - 1-Add new employee
 - 2-Print employee data
 - 3-remove employee from the system

- Write script to send an email for your friend when your device is on with its location

Start on hackerRank

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

```
1 45 def Respond(voice_data):
2
3 46     if 'الاسم' in voice_data or 'اسم' in voice_data :
4
5 47         Bixby_Speak('الزعيم معتصم وصل يا رجاله')
6
7 48         # Bixby_Speak('Moatasem Big Boss')
8
9 49     if 'الوقت' in voice_data or 'الساعة' in voice_data :
10
11 50         Bixby_Speak(ctime())
12
13 51     if 'بحث' in voice_data or 'البحث' in voice_data :
14
15 52         search = record('العبد الله كفاية بس جوجل حكاية')
16
17 53         # search = record('what dow want to search')
18
19 54         url = 'https://google.com/search?q=' + search
20
21 55         webbrowser.get().open(url)
22
23 56         # Bixby_Speak('Here is what i Found For' + search)
24
25 57         Bixby_Speak('خلصانه بشياكه' + search)
```