Moatasem Elsayed

Bio

- Senior 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

Introduction

- history
- versions
- Interpreter vs compiler
- Start program
- comments
- Python CMD
- String
- Lab1 (print)
- Variables
- Statically vs Dynamically types
- Comma operator
- Lab2 (arithmetic equations)
- Data Types

- Input
- Operators
- If
- For

Introduction To Python

* Python is a popular programming language created by Guido van Rossum in 1991

* Python works on different platforms (Windows, Mac, Linux) and it has a simple
    syntax similar to the English language

* The most recent major version of Python is Python 3
Python 2

Python Versions

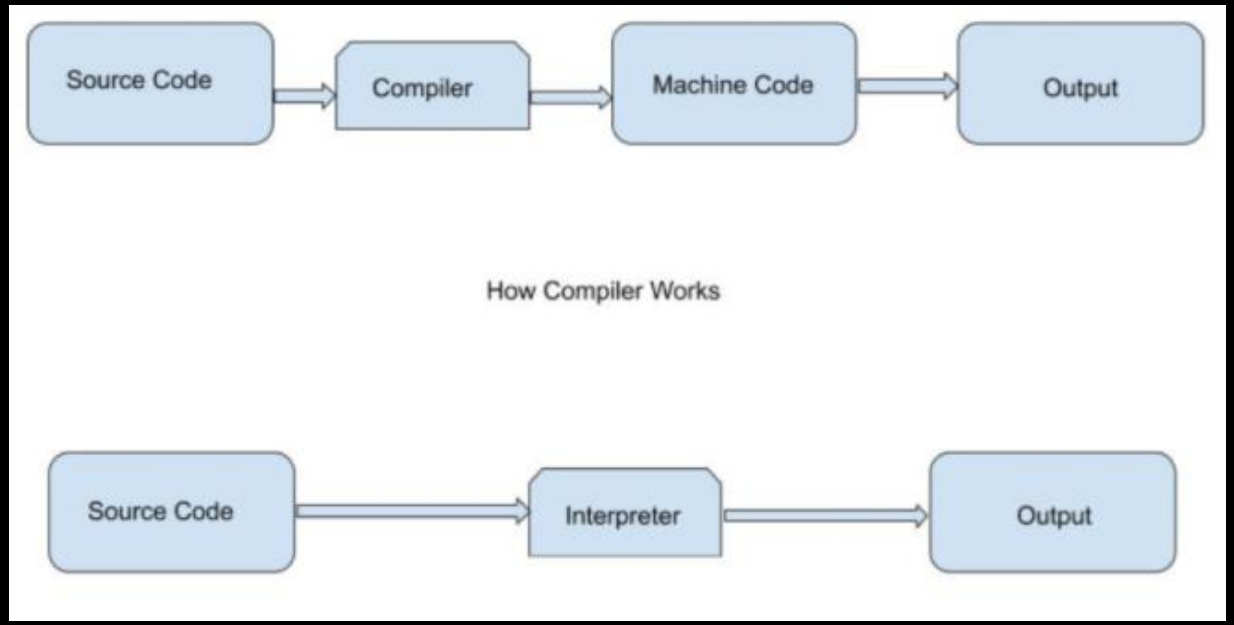
* After installation of python and add to path variable
* Open cmd and write this command
  Python --version

```
C:\Users\engmo>python --version
Python 3.9.7
```

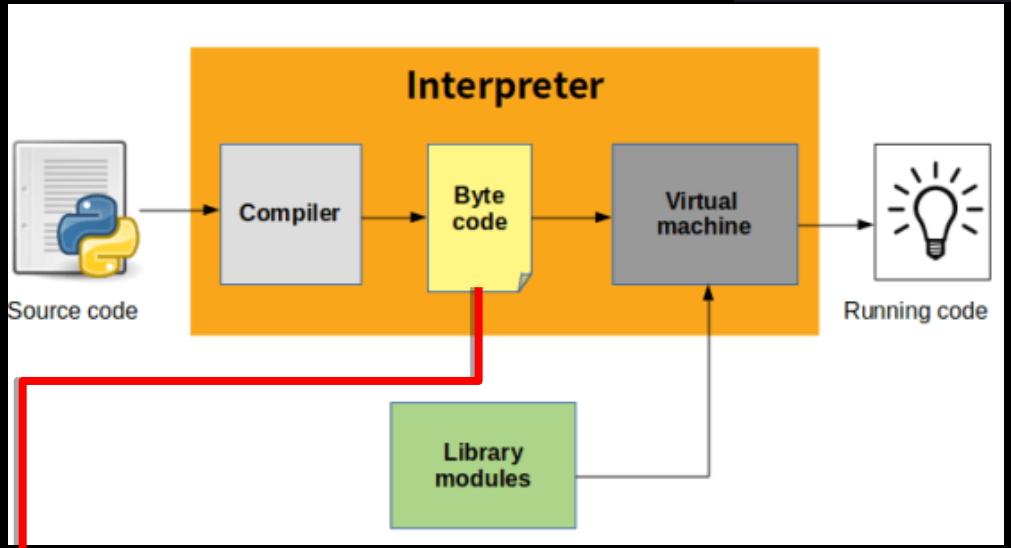
```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~$ python3 --version
Python 3.10.6
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~$
```

Interpreter Vs Compiler

Python is an interpreted programming language not compiler



Python Interpreter




```
01_introduction > test > test.py > ...
1  x = 10
2  print(x)
3
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SERIAL MONITOR

moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction$ python3 -m dis test/test.py
  1           0 LOAD_CONST               0 (10)
              2 STORE_NAME               0 (x)

  2           4 LOAD_NAME                1 (print)
              6 LOAD_NAME                0 (x)
              8 CALL_FUNCTION            1
             10 POP_TOP
             12 LOAD_CONST               1 (None)
             14 RETURN_VALUE
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction$
```

How to Run interpreting ?

test.py

01_introduction > test > test.py

```
1  print("hello world")
2
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIAL MONITOR

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ python3 test.py
hello world
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

OR

test.py

01_introduction > test > test.py

```
1  #!/usr/bin/python3
2  print("hello world")
```

Shebang

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIAL MONITOR

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ which python3
/usr/bin/python3
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ chmod u+x test.py
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ ./test.py
hello world
```

Comments

```
#this is comment
print("hello")

'''
Multi Line Comments
'''
```

Comments :
1- # for single lin

2- ' ' '
Multi lines
' ' '

1- No Semicolons

2- Internal functions have don t need a library to include

3- space is very important

Strings

test.py

01_introduction > test > test.py

```
1  #!/usr/bin/python3
2  print("hello world")
3  print('hello world')
4  print('''hello world''')
5
```

```
hello world
hello world
hello world
```

```
#this is good because :
print("hello mr's Moatasem")
print('he said to "Moatasem" hi mr ')
```

output

```
hello mr's Moatasem
he said to "Moatasem" hi mr
```

Quick Task (print)

Write a python code that print your information.
Full Name, Birth Year, Faculty, E-mail, Address

```
Full Name: Moatasem Elsayed
Birth:21/3/1994
Faculty:HTI
E-mail:eng.moatasem.9@gmail.com
Address:Giza
```

```
print("Full Name: Moatasem Elsayed\nBirth:21/3/1994\nFaculty:HTI\nE-mail:eng.moatasem.9@gmail.com\nAdd
```

Variables

Unlike other programming languages, Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.

```
x=5
print(type(x))
x="ahmed"
print(type(x))
x=3.5
print(type(x))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
```

Statically vs Dynamically types

Statically typed languages

A language is statically typed if the type of a variable is known at compile time. For some languages this means that you as the p pecify what type each variable is; other languages (e.g.: Java, C, C++)

```
1 main.cpp +
int x =10;
```

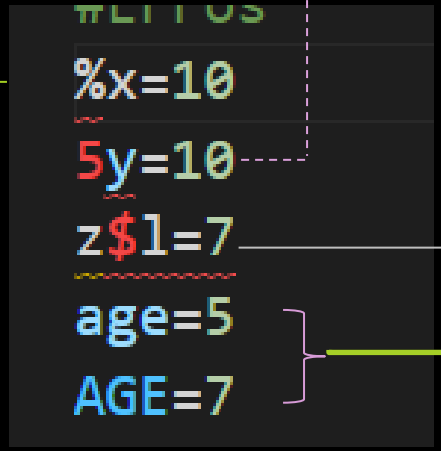
Dynamically typed languages

A language is dynamically typed if the type is associated with run-time values, and not named variables/fields/etc. Examples: Perl, Ruby, Python, PHP, JavaScript, Erlang

```
#!/usr/bin/python3
x = 10  # int
x = "hello"  # str
x = [1, 1.5, "hello"]  # list
```

Variable naming rules

A variable starts with a letter or the underscore (_)character
A variable cannot start with a number
A variable only contains alphanumeric characters and underscores (A z, 0 9 , and _)
Variables are case sensitive (age, Age and AGE are three different variables)



Comma Operator

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

```
x,y,z=1,2,3
print(x)
print(y)
print(z)
```

```
1
2
3
```

Same concept in c++14

```
mytuple = std::make_tuple (10, 2.6, 'a');

std::tie (myint, std::ignore, mychar) = mytuple;
```

Quick task Arithmetic

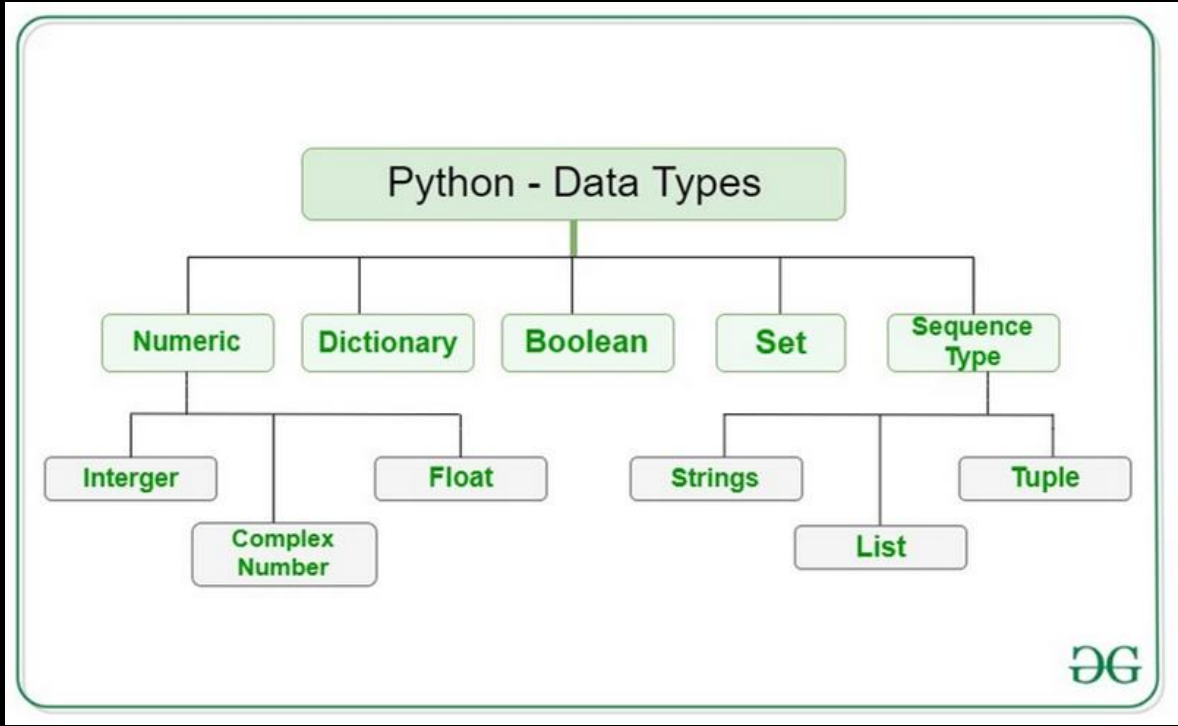
Print sum of two numbers

Print sub of two numbers

Print Div of two numbers

Print Mult of two numbers

Data Types


Python - Data Types
Numeric
Dictionary
Boolean
Set
Sequence Type
Interger
Float
Complex Number
Strings
Tuple
List

Data Types

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set
Boolean Type:	bool

```
x="Moatsem"
print(x)
print(type(x))
```

```
Moatsem
<class 'str'>
```

```
#complex
x=5+6j
print(x)
print(type(x))
```

```
(5+6j)
<class 'complex'>
```

```
#list
x=[1,2,3,4,5]
print(x)
print(type(x))
```

```
[1, 2, 3, 4, 5]
<class 'list'>
```

Tuple =const list

```
x=(1,2,3,4,5)
print(x)
print(type(x))
```

```
(1, 2, 3, 4, 5)
<class 'tuple'>
```

```
##################
x=True
print(type(x))
print(x)
```

```
<class 'bool'>
True
```

bool

```
1  print(10 > 9)
2  print(bool("Hello"))
3  print(bool(15))
4  print(10 > 9)
5  print(1 > 2)
6  print(bool("Hello"))
7  print(bool(15)     )
8  print(bool(False)  )
9  print(bool(None)   )
10 print(bool(0)      )
11 print(bool("")     ) #empty string
12 print(bool(())     ) #empty tuple
13 print(bool([])     ) #empty list
14 print(bool({})     ) #empty set
```

```
True
True
True
True
False
True
True
False
False
False
False
False
False
False
False
```

list

Collection - ordered - **mutable**

slicing

```
1
2
3
```

```
x=["moatasem",1,2.5,3,4,5,8,9,]
x[1]=55
print(x[0])
print(x[1])
print(x[1:3])
print(x[-1])
print(x[-2])
print(x[2:])
print(x[1:4:2])
```

```
moatasem
55
[55, 2.5]
9
8
[2.5, 3, 4, 5, 8, 9]
[55, 3]
```

```
hello
moatsem@moatsem-Idea
s="hello world"
echo ${s:2:3}
echo "${s::-5}"
moatsem@moatsem-Idea
llo
hello
moatsem@moatsem-Idea
```

```
// Customer Class
class Customer {

    // class Variables
    string name;
    mutable string placedorder;
    int tableno;
    mutable int bill;

    // member methods
public:


    // constructor
    Customer(string s, string m, int a, int p)
    {
        name= s;
        placedorder=m;
        tableno = a;
        bill = p;
    }

    // to change the place holder
    void changePlacedOrder(string p) const
    {
        placedorder=p;
    }
```

Tuple

```
Traceback (most recent call last):
  File "D:\Embedded System\Embedded Linux\My presentation\0
    x[1]=55
TypeError: 'tuple' object does not support item assignment
```

Collection ordered unchangeable
(Immutable)

```
1
2
3  x=("moatasem",1,2.5,3,4,5,8,9)
4  #x[1]=55
5  print(x[0])
6  print(x[1])
7  print(x[1:3])
8  print(x[-1])
9  print(x[-2])
1  print(x[2:])
1  print(x[1:4:2])
1
14
```

```
moatasem
1
(1, 2.5)
9
8
(2.5, 3, 4, 5, 8, 9)
(1, 3)
```

Dictionary

```python
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
print(type(thisdict))
print(thisdict)
print(thisdict.keys())
print(thisdict.values())
print(thisdict["brand"])
print(len(thisdict))
```

```
<class 'dict'>
{'brand': 'Ford', 'electric': False, 'year': 1964, 'colors': ['red', 'white', 'blue']}
dict_keys(['brand', 'electric', 'year', 'colors'])
dict_values(['Ford', False, 1964, ['red', 'white', 'blue']])
Ford
4
```

Json


```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

No Duplicate Member

python

```python
#!/usr/bin/python3
dictt = {
    "ID": 123,
    "name": "Moatasem",
    "Email": "eng.moatasem.9@gmail.com",
    "Email": "eng.moatasem.8@gmail.com",
}

print(dictt)
```

```
{'ID': 123, 'name': 'Moatasem', 'Email': 'eng.moatasem.8@gmail.com'}
```

c++

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ cat main.cpp
#include <iostream>
#include <map>

int main() {

  std::map<std::string, std::string> values = {
      {"ID", "123"},
      {"Name", "Moatasem"},
      {"email", "eng.moatasem.9@gmail.com"},
      {"email", "eng.moatasem.8@gmail.com"},
  };

  for (auto i : values) {
    std::cout << i.first << ": " << i.second << std::endl;
  }
  std::cout << std::endl;
}
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ ./a.out
ID: 123
Name: Moatasem
email: eng.moatasem.9@gmail.com

moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

set

items are unordered, changeable(mutable), and do not allow duplicate values.

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
print(type(thisset))
```

```
{'apple', 'banana', 'cherry'}
<class 'set'>
```

```
#!/usr/bin/python3
st = {1, 2, 2, 2, 2, 2, 6, 2, 7, 2, 8, 6}
print(st)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SERIAL MONITOR

moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ python3 test.py
{1, 2, 6, 7, 8}
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

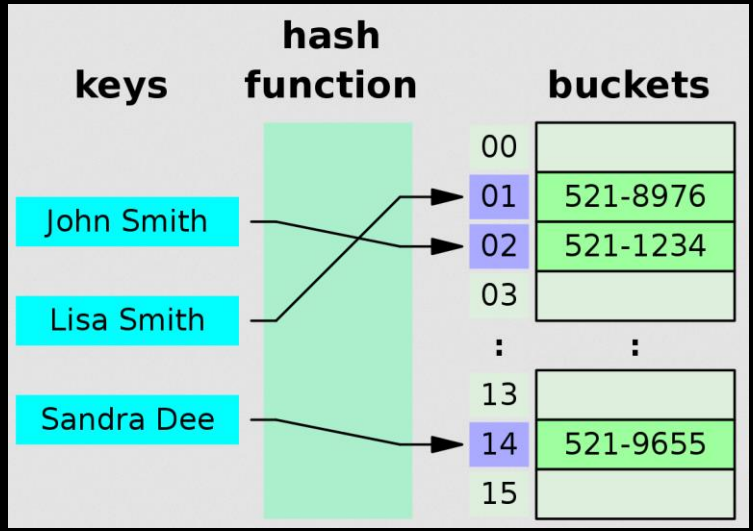
Set using Hash table

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ ./a.out
8,51,10,62,400,3,9,-7,22,11,
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ cat main.cpp
#include <iostream>
#include <unordered_set>


int main(){

    std::unordered_set<int> values={11,22,3,400,51,62,-7,8,9};

    values.insert(values.end(),10);

    for (int i : values){
        std::cout <<i<<",";
    }
    std::cout <<std::endl;

}

moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

Hash table

```python
import time

# Creating a list of numbers
numbers_list = list(range(1, 1000001))

# Creating a set of numbers
numbers_set = set(numbers_list)

# Searching for a number in a list
start_time = time.time()
print(1000000 in numbers_list)
end_time = time.time()
print("Time taken to search in list:", end_time - start_time, "seconds")

# Searching for a number in a set
start_time = time.time()
print(1000000 in numbers_set)
end_time = time.time()
print("Time taken to search in set:", end_time - start_time, "seconds")
```

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentatior
True
Time taken to search in list: 0.0038139820098876953 seconds
True
Time taken to search in set: 2.1457672119140625e-06 seconds
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentatior
```



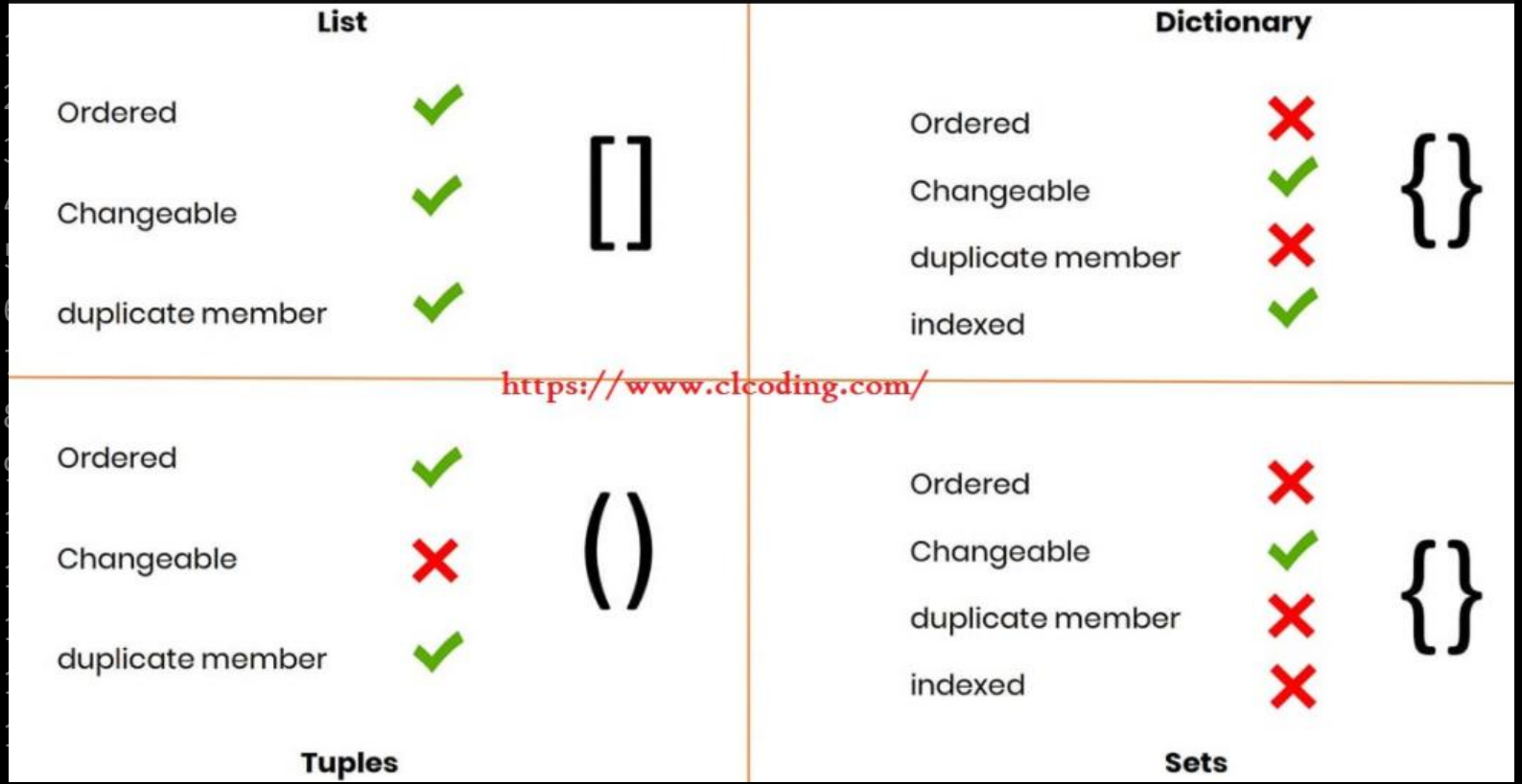
Range

```
1  for i in range(5):
2      print(i,end=",")
3
4  print("")
5
6  for i in range(0,4,2):
7      print(i,end=",")
8  print("")
9  for i in range(1,5,1.5)
10     print(i)
11
12
13 x=list(range(0,10))
14 print(x)
```

```
0,1,2,3,4,
0,2,
Traceback (most recent call last):
  File "D:\Embedded System\Embedded Linux\My presentation\01p
    for i in range(1,5,1.5):
TypeError: 'float' object cannot be interpreted as an integer
```

```
0,2,
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Main Comparison



Convert string to list

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

```
name="moatasem"
```

output

```
PS D:\Embedded System\Embedded Linux\My pr
['m', 'o', 'a', 't', 'a', 's', 'e', 'm']
```

```
name="moatasem"
print(list(name))
```

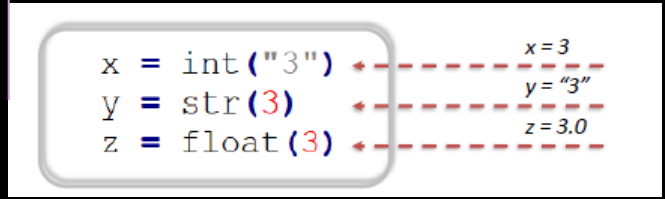
Input

```
print('Enter your name:')
name = input()
print('Hello, ' + name)
```

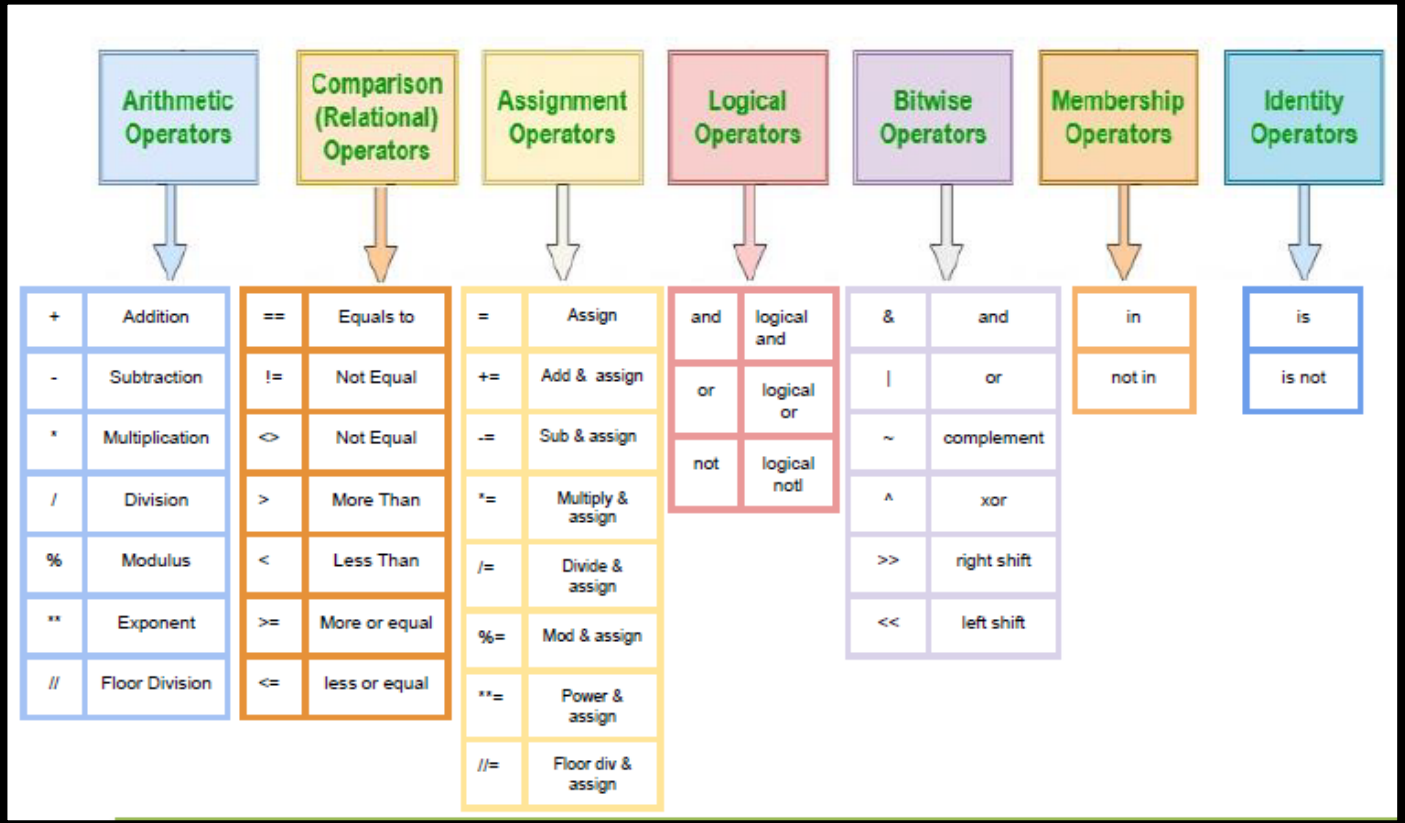
```
PS D:\Embedded System
Enter your name:
moatasem
Hello, moatasem
```

```
print('Enter your Age:')
AgE = int(input())
print('My age is :{}'.format(AgE) )
```

```
Enter your Age:
12
My age is :12
```



Operators


Arithmetic Operators
+ Addition
- Subtraction
* Multiplication
/ Division
% Modulus
** Exponent
// Floor Division
Comparison (Relational) Operators
== Equals to
!= Not Equal
<> Not Equal
> More Than
< Less Than
>= More or equal
<= less or equal
Assignment Operators
= Assign
+= Add & assign
-= Sub & assign
*= Multiply & assign
/= Divide & assign
%= Mod & assign
**= Power & assign
//= Floor div & assign
Logical Operators
and logical and
or logical or
not logical notl
Bitwise Operators
& and
| or
~ complement
^ xor
>> right shift
<< left shift
Membership Operators
in
not in
Identity Operators
is
is not

Arithmetic

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

b2.py > ...

```
1  x=3
2  y=2
3  print(x**y)
4  print(x//y)
```

```
9
1
```

Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Assignment Operators

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

```
2  val1=12
3  val2=14
4  if val1>10 && val2 >10 :
5      print("hello")
```

ERROR

```
val1=12
val2=14
if val1>10 and val2 >10 :
    print("hello")
```

RUN

Alternative tokens

There are alternative spellings for se
the language, each alternative token
stringification operator can make the
Despite being four-letters long, `%:%:`

Primary	Alternative
&&	and
&=	and_eq
&	bitand
|	bitor
~	compl
!	not
!=	not_eq
||	or
|=	or_eq
^	xor
^=	xor_eq
{	<%
}	%>
[	<:
]	:>
#	%:
##	%:%:

C++

test.py | main.cpp 1

01_introduction > test > main.cpp > ...

```
1  #include <iostream>
2
3  int main() { std::cout << (1 and 1) << std::endl; }
4
```

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ cat main.c
#include <iso646.h>
#include <stdio.h>

int main(){
printf("%d\n" , 1 and 1 );
}
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ ./a.out
1
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

C

Bitwise(with mentor)

```
a = 60              # 60 = 0011 1100
b = 13              # 13 = 0000 1101
c = 0
c = a & b;          # 12 = 0000 1100
print ("Line 1 - Value of c is ", c)
c = a | b;          # 61 = 0011 1101
print ("Line 1 - Value of c is ", c)
c = a ^ b;          # 49 = 0011 0001
print ("Line 1 - Value of c is ", c)
c = ~a;             # -61 = 1100 0011
print ("Line 1 - Value of c is ", c)
c = a << 2;         # 240 = 1111 0000
print ("Line 1 - Value of c is ", c)
c = a >> 2;         # 15 = 0000 1111
print ("Line 1 - Value of c is ", c)
```

Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

01_introduction > test > test.py

```
1  #!/usr/bin/python3
2
3  print(1 in [1, 2, 3, 4, 5, 6, 7])
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIAL MONITOR

```
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ python3 test.py
True
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

Identity Operators (not too much usable but good to know)

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

```
2
3  a = 10
4  b = 10
5  print(id(a))
6  print(id(b))
7  if a is b:
8      print(" a is c")
9  if a == b:
10     print(" a equals to c")
11
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SERIAL MONITOR
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$ python3 test.py
140192221495824
140192221495824
 a is c
 a equals to c
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/My presentation/01python/01_introduction/test$
```

```
1  #!/usr/bin/python3
2
3  a = [10]
4  b = [10]
5  print(id(a))
6  print(id(b))
7  if a is b:
8      print(" a is c")
9  if a == b:
10     print(" a equals to c")
11
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SERIAL MO
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma
140670812824640
140670812856320
 a equals to c
moatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma
```

If condition

```
1  a = 200
2  b = 33
3  if b > a:
4    print("b is greater than a")
5  elif a == b:
6    print("a and b are equal")
7  else:
8    print("a is greater than b")
9
10
```

```
PS D:\Embedded System\Embedded Linu
a is greater than b
PS D:\Embedded System\Embedded Linu
```

Shorthand If

```
11
12   a = 200
13   b = 33
14   if a > b: print("a is greater than b")
```

Shorthand If ... Else

```
14  a = 2
15  b = 330
16  print("A") if a > b else print("B")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2> python .\lab3.py
B
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2>
```

Ternary

```
19  a = 330
20  b = 330
21  print("A") if a > b else print("=") if a == b else print("B")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2> python .\lab3.py
=
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2>
```

Nested if

```
23 ########################################
24 x = 41
25 if x > 10:
26     print("Above ten,")
27     if x > 20:
28         print("and also above 20!")
29     else:
30         print("but not above 20.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded Linux\My presentation
Above ten,
and also above 20!
PS D:\Embedded System\Embedded Linux\My presentation
```

pass Statement

```
1  a = 33
2
3  b = 200
4  if b > a:
5
6          pass
7
8
9
10
11
12
13
14
```

Like empty bracket

Quick Task(2)

Write a python code that handle the following login system:

User Name	Password
Ahmed	1394
Ali	6078
Amr	9345

If the data entered is correct, the system shall show a welcome message, if not the system will print incorrect entry.

loop

While Loop

```
32  ########################
33  i=0
34  while i<5:
35      print(i)
36      i+=1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
0
1
2
3
4
```

for

```
38  for i in range(5):
39      print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded
0
1
2
3
4
```

Break vs continue

```
41  ##########################
42  for i in range(1,10):
43      if i %2 == 0:
44          break
45      print(i," odd")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded Linux
1  odd
PS D:\Embedded System\Embedded Linux
```

```
41  ##########################
42  for i in range(10):
43      if i %2 == 0:
44          continue
45      print(i,"odd")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded Linux
1 odd
3 odd
5 odd
7 odd
9 odd
```

Else in For Loop

```
47
48 for x in range(6):
49     print(x)
50 else:
51     print("Finally finished!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
2
3
4
5
Finally finished!
```

Nested Loops

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

Shorthand for

```
51  ##################################################################333
52  [print(i) for i in range(3) if i%2==0]
53  [print(i) for i in "moatasem" ]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2> python .\lab3.py
0
2
m
o
a
t
a
s
e
m
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 2>
```

Reverse string

Write a python code that ask the user to enter a sentence and then print it in opposite direction

```
please enter name moatasem
mesataom
```

cstyle

```
#####################################
name=input("please enter name")
for i in range(len(name)):
    print(name[-1-i],end="")
```

Python style

```
#another solution
txt = name[::-1]
print(txt)
```

Import modules

test.py > ...

```python
import psutil

# Get CPU usage percentage
print("CPU Usage:", psutil.cpu_percent())

# Get memory usage statistics
memory = psutil.virtual_memory()
print("Total Memory: ", memory.total/1000000000, "G")
print("Available Memory:", memory.available/1000000000, "G")
print("Used Memory:", memory.used/1000000000, "G")
print("Memory Usage Percentage:", memory.percent)

# Get disk usage statistics
disk = psutil.disk_usage('/')
print("Total Disk Space:", disk.total/1000000000, "G")
print("Used Disk Space:", disk.used/1000000000, "G")
print("Free Disk Space:", disk.free/1000000000, "G")
print("Disk Usage Percentage:", disk.percent)

```

```
Disk Usage Percentage: 61.3
● moatsem@moatsem-IdeaPad-Gaming-3-15IAH7
CPU Usage: 0.0
Total Memory:  16.475721728 G
Available Memory: 3.85619968 G
Used Memory: 9.902964736 G
Memory Usage Percentage: 76.6
Total Disk Space: 502.392610816 G
Used Disk Space: 292.352561152 G
Free Disk Space: 184.444678144 G
Disk Usage Percentage: 61.3
● moatsem@moatsem-IdeaPad-Gaming-3-15IAH7
```

13

14

Favourite Folder

test.py > ...

```python
#!/usr/bin/python3
import os


favoriteFolder = [
    "/home/moatsem/temppoky/poky",
    "/home/moatsem/Diploma/mypresetation",
    "/home/moatsem/c++/workspace"
]

val = int(input("please select your dir (index start with 0): "))

os.popen(r"nautilus {} ".format(favoriteFolder[val]))

```

Time Now

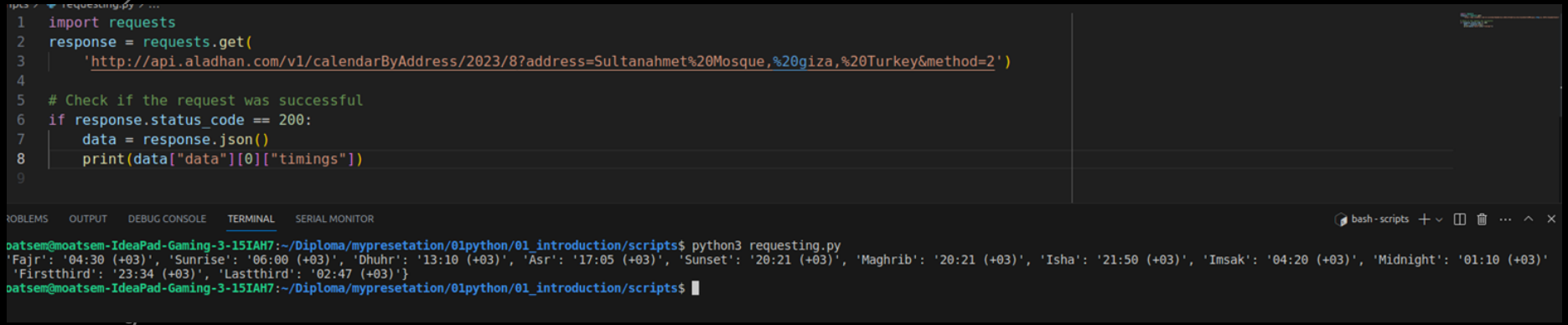
```
1
2   import datetime
3   now = datetime.datetime.now()
4   print ("Current date and time : ")
5   print (str(now)[0:-7])
6
7
8
9
10  Current date and time :
11  2021-10-23 14:04:49
12
13
14
```

pyfiglet

```python
# pip install pyfiglet
import pyfiglet
result = pyfiglet.figlet_format("Moatasem Elsayed")
print(result)

```

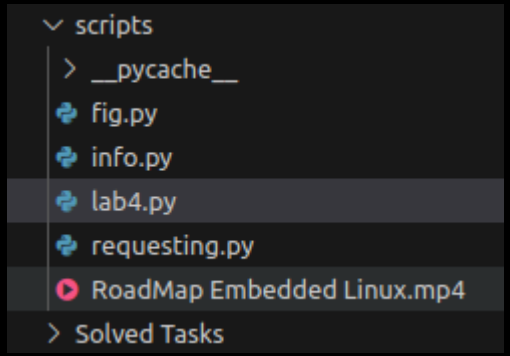
```
 __  __                 _
|  \/  | ___   __ _ ___| |_ __ _ ___  ___ _ __ ___
| |\/| |/ _ \ / _` / __| __/ _` / __|/ _ \ '_ ` _ \
| |  | | (_) | (_| \__ \ || (_| \__ \  __/ | | | | |
|_|  |_|\___/ \__,_|___/\__\__,_|___/\___|_| |_| |_|


 _____ _                                 _
| ____| |___  __ _ _   _  ___  __| |
|  _| | / __|/ _` | | | |/ _ \ _` |
| |___| \__ \ (_| | |_| |  __/ (_| |
|_____|_|___/\__,_|\__, |\___|\__,_|
                   |___/
```

requests


import requests
response = requests.get(
'http://api.aladhan.com/v1/calendarByAddress/2023/8?address=Sultanahmet%20Mosque,%20giza,%20Turkey&method=2')
# Check if the request was successful
if response.status_code == 200:
data = response.json()
print(data["data"][0]["timings"])
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
SERIAL MONITOR
bash - scripts
oatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/mypresetation/01python/01_introduction/scripts$ python3 requesting.py
'Fajr': '04:30 (+03)', 'Sunrise': '06:00 (+03)', 'Dhuhr': '13:10 (+03)', 'Asr': '17:05 (+03)', 'Sunset': '20:21 (+03)', 'Maghrib': '20:21 (+03)', 'Isha': '21:50 (+03)', 'Imsak': '04:20 (+03)', 'Midnight': '01:10 (+03)'
'Firstthird': '23:34 (+03)', 'Lastthird': '02:47 (+03)'}
oatsem@moatsem-IdeaPad-Gaming-3-15IAH7:~/Diploma/mypresetation/01python/01_introduction/scripts$

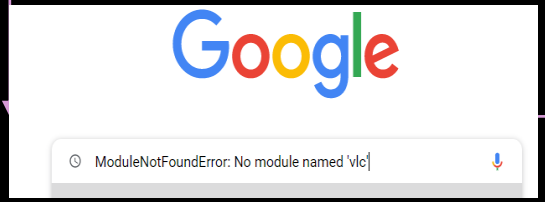
Youtube Downloader

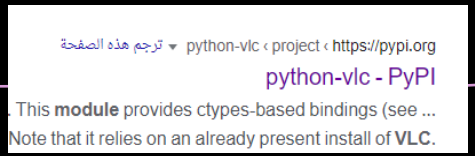
```python
# pip install pytube
from pytube import YouTube
YouTube('https://www.youtube.com/watch?v=WHTjqH1L7_c&list=PLkH1REggdbJojAFRPJbq4nXKLTnytNzw2&index=1&t=16s')\
    .streams.filter(progressive=True, file_extension='mp4').first().download()
```

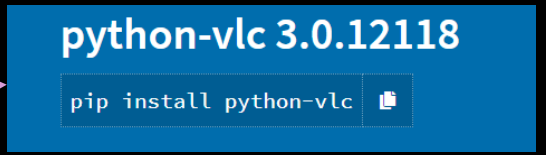


Install Module

```
PS D:\Embedded System\Embedded Linux\My presenta
Traceback (most recent call last):
  File "D:\Embedded System\Embedded Linux\My pre
    import vlc
ModuleNotFoundError: No module named 'vlc'
PS D:\Embedded System\Embedded Linux\My presenta
```

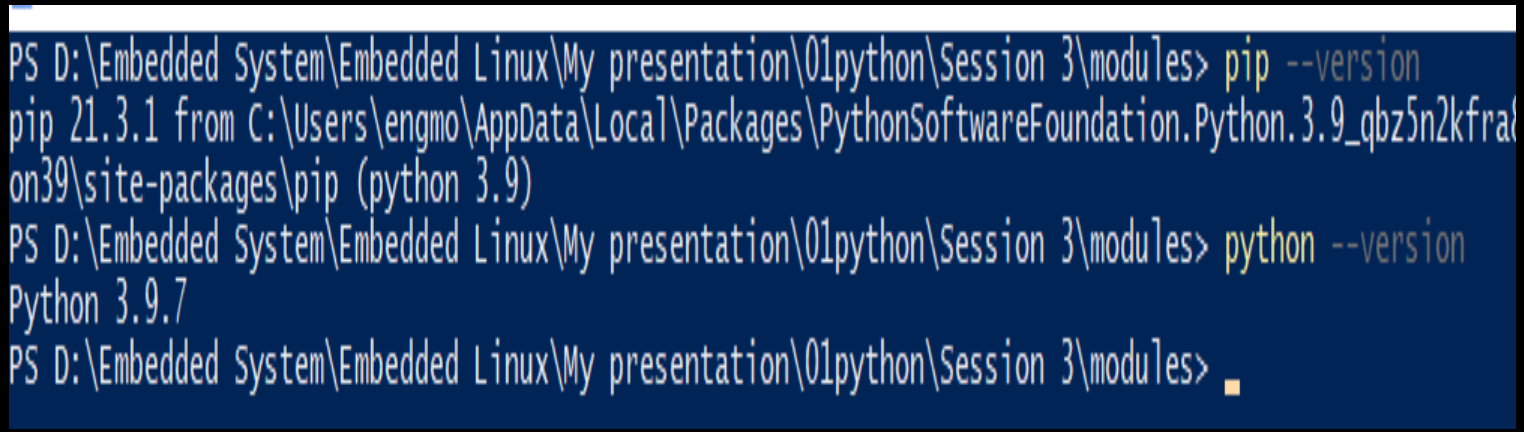







```
PS D:\Embedded System\Embedded Linux\My presentation\01python\Session 1\lab3> pip install python-vlc
Collecting python-vlc
  Downloading python_vlc-3.0.12118-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 1.1 MB/s
Installing collected packages: python-vlc
Successfully installed python-vlc-3.0.12118
```

Big Problem (Take Care)



Convert text to speak

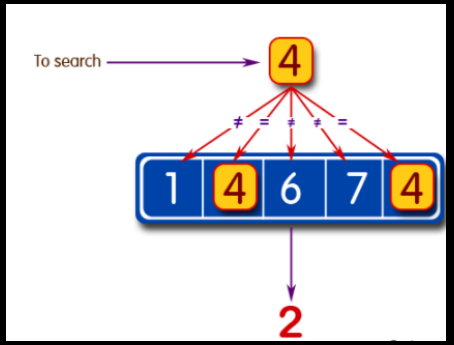
ipts > convertTextToSpeak.py > ...

```python
# pip install gTTS
# pip install  python-vlc
from gtts import gTTS
import vlc
myobj = gTTS(text='صباح الفل يا كبير', lang='ar', slow=False)
# Saving the converted audio in a mp3 file named
myobj.save("welcome.mp4")
# Playing the converted file
p = vlc.MediaPlayer("./welcome.mp4")
p.play()
while True:
    pass

```

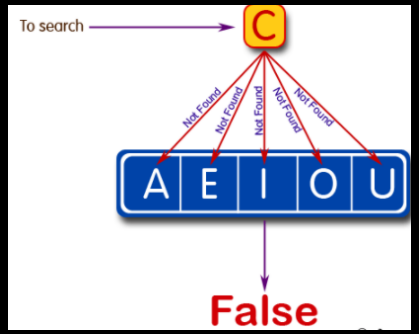
sudo apt-get install vlc

Tasks

1. Write a Python program to count the number 4 in a given list.



2. Write a Python program to test whether a passed letter is a vowel or not.



3. Write a python program to access environment variables.

PATH

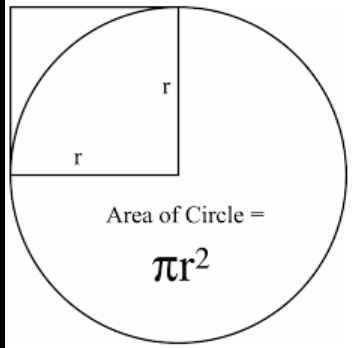
os

Tasks(2)

Write a Python program which accepts the radius of a circle from the user and compute the area.

Python: Area of a Circle

In geometry, the area enclosed by a circle of radius r is πr2. Here the Greek letter π represents a constant, approximately equal to 3.14159, which is equal to the ratio of the circumference of any circle to its diameter.



Tasks(3)

Print the calendar of a given month and year

```
Input the year : 2017
Input the month : 04
     April 2017
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Python calendar.month(theyear, themonth, w=0, l=0):

The function returns a month's calendar in a multi-line string using the formatmonth() of the TextCalendar class.

'l' specifies the number of lines that each week will use.