introduction

```
# What is python?
# - Programming language can do anything.
# - Free and open source.
# - Interpreted.
# - Interactive. (write the code and while running the output appear in the terminal).


# Why python?
# - Easy to install.
# - Clean and easy to learn.
# - Error handling.
# - Debugging is easy (debugger is created with python).
# - Cross platform (run on any operating system).
# - Expressive.
# - Support OOP.
# - Integrated (can create app with more than one programming language).
# - Support module packages.
# - Large set of lists and plugins.
# - Memory management (garbage collection).
# - Multi-purpose.
# - Great community.
# - Growing fast.
# - You can switch careers with the basic knowledge.


# What’s python used for?
# - Web development (Django, flask) .
# - Games (PyGame).
# - Desktop Apps (pyGUI, tkinter) .
# - Hacking.
# - Machine learning & data science.
# - AI & Robots.
# - Automation.
# - Web scrapping (harvest).
# - Android game.


# Apps creates with Python!
# - Disqus.
# - Instagram.
# - Spotify.
# - Dropbox.
# - Uber.


# Companies using python!
# - Google.
# - Facebook.
# - Netflix.
```

---

02 What I Need?

```
# - Text editor or IDE → VS code or PyCharm.
# - Download Python.
# - Python extension.
# - Learn command line.
# - Download Cmder.
# - Knowledge in programming.
# - Version.
# - Path knowledge.
```

---

03 syntax and first App

```
print('i love python')  # used to print to the screen
print("i love programming") # used to print to the screen
print("hello"); print("world")  # semicolon used to print multiple command in one
line

# python depends on indentation

# true syntax
if True:
    print("hello world")


# false syntax
if True:
print("hello world")
```

---

04 Comments

```
#  hash: used to make comment
# used to make multiple
# line comment

# ==> Why Use Comments
# - Information about the file.
# - Who created the file.
# - When the file created.
# - Why the file is created.
# - Comment can written before the command line.
# - Comment can written beside the command line.
# - Prevent code from run.
# - Comment should be useful.

# - Note That of The Following:
"""this isn't
multiple line comment
this is unassigned text/string"""
```

---

05 dealing with data

```
# - Our Apps contains code + data.
# - Code is the lines you write to manage data and dealing with this data.
# - To structure The Data we need to categorize [number, string, Boolean]
# - Number → student age, student phone number, teacher salary, …
# - String → student name, teacher name, event name, …
# - Booleans → teacher got salary or not, student age is suitable or not, …
# - Data is stored in computer memory.
# - We use variable to refer to this data.
# - Variable not containing the data its only refers to its location memory.
# - Code is using the data to perform operation [add, edit, delete].
```

---

06 some data types

```
# type()     # is a built in function to print the type of the data .
# → all data in python is an object.
print(type(10)) # int ➔ integer .
print(type(-10)) # int ➔ integer .
print(type(100.11)) # float ➔ floating point number .
print(type(-10.7835)) # float ➔ floating point number .
print(type("hello world")) # str ➔ string .
print(type([1,2,3,4,5])) # list ➔ List
print(type((1,2,4,5))) # tuple ➔ Tuple .
print(type({'one': 1, "two":2})) # dict ➔ dictionary .
print(type(2 == 4)) # bool ➔ Boolean .
```

---

07 variables part 1

```
# Syntax:
# [variable name] [assignment operator] [value]

# Example:
myVariable = 'my value'
print(myVariable)

# → name convention and rules:
# - can start with (a-x, A-Z) or underscore.
# - you can't start with numbers or special characters.
# - can include (0-9) or underscore.
# - can't include special character.
# - python is case sensitive.
# - Name is not like name.
# - you can't use the variable before assign it (assign then use it).

# ➔ Naming the Variables:
# [1] name = 'osama'              # single word ➔ normal case.
# [2] myName = 'Osama Elzero'     # two words ➔ camel case.
# [3] my_name = 'Osama Elzero'    # two words ➔ sanke_case.
```

---

08 Variables part 2

```
# ❖ source code: original code you write it in computer.
# ❖ translation: converting source code into machine language.
# ❖ compilation: translate code before run time.
# ❖ run-time: period App take to executing command.
# ❖ interpreted: code translated on the fly during execution.

# → the following are reserved keywords in python:
break, for, in, not, try, except, class, is, while, lambda, break, ...

# - to know all keywords →
help("keywords")

# - to assign more than one variable in one line:
a, b, c = 1 ,2 ,3
```

---

09 Escape Sequences

```
# - \b # escape back space.
print("hello \b world")

# - \t # escape horizontal tab space.
print('hello \t world')

# - \n # escape line feed.
print('hello \n world')

# - \ # escape new line.
# Example :--
print('hello \
I Love \
Python')

# - \\ # escape back slash.
print('hello \\ world')

# - \’ # escape single quote.
print('hello \'world\'')

# - \” # escape double quote.
print('hello \"world\"')

# - \r # carriage return
# Example:
print("123456\rABCD")   # means take characters after “\r” to replace them in
# the chars before \r and print them.
# the output is → ABCD56
# - \xhh # character hex value.
```

---

10 Concatenation

```
msg = "I Love Python\n"
lang = "Hello World"
print(msg+lang) # we concatenate using plus “+”

# - can only concatenate string with string.
# - can’t concatenate string with int, float, …
```

---

11 strings

```
# - Strings can written in single quote or double quote.
# - Can write single quote in double quote
# Example:
print("I Love 'Python' ") # to unused escape sequence .


# - Can write double quote in single quote.
# Example:
print('I Love "Python" ') # to unused escape sequence .

# - To make string with multiple line use “”” multiple line “”””
# Example:
print('''
hello
world
I Love
Python programming
''') # or using triple single quote """ """
```

---

12 String indexing and slicing

```
# - All data in python is an object.
# - Object contain elements.
# - Every element has its own index.
# - Python use zero based indexing (index start from zero).
# - Sue square bracket to access the elements.
# - Enable accessing parts of strings, tuples, lists, …

# → indexing (access single item):
# Syntax:
# print(str_name([index]))

myString = 'I Love python'
print(myString[5]) # output is : e
print(myString[9]) # output is : t
```

```
print(myString[-1]) # output is : n
# -1 means first character from end.
# Negative value refers to start from in the end.

# →slicing (access multiple sequence items):
# [start:end] # end not included
# [start:end:step]
# [:6] # if start is not here will start from zero.
# [6:] # If end is not here will go to the end.
# [:] # if start, end is not here will print all data.

message = "osama mohamed ahmed"
print(message[:])       # osama mohamed ahmed
print(message[1:])      # sama mohamed ahmed
print(message[:5])      # osama
print(message[:5:2])    # oaa
```

---

13 strings methods part 1

```
# len() # is a built in function that return the number of the string , int , data , …

print('hello world') # output is 11
# Note that spaces count in the string.

# 1- strip() rstrip() lstrip() .
# strip() : remove spaces in the string in the right , left side.
# rstrip() : remove spaces in the string that exist in the right side.
# lstrip() : remove spaces in the string that exist in the left side.

# example :
name = '  osama  '
print(name.strip())

# - if put character will remove character instead of the spaces
# Example
name = '##osama##'
print(name.strip('#'))


# 2- title()
# make first letter from all words capital, and the letters after the numbers capital.

a = 'I Love 2d Graphics and 3g technology and python'
print(a.title()) # output is : I Love 2D Graphics And 3G Technology And Python


# 3- capitalize()
# make only first letter in the first word capital
a = 'I Love 2d Graphics and 3g technology and python'
print(a.capitalize())   #output is : I love 2d graphics and 3g technology and python
```

```
# 4- zfill()
# o that fill numbers by adding zeros to numbers, that take the maximum numbers.
# the variable should be in double / single quote (present variable as a string).
# ➔ Note That The value of the variable should be string.
# example:
a, b, c = '1', '2', '3'
print(a.zfill(3))
print(b.zfill(3))
print(c.zfill(3))


# 5- upper()
# that make letters in upper case (capital all letters)
my_name = 'osama mohamed'
print(my_name.upper())

# 6- lower()
# that make all letters in lower case (small letters)
myName = "OSAMA MOHAMED"
print(myName.lower())
```

---

14 strings methods part 2

```
# 7- split() rsplit()
# split(): that means split the string into “ list “ using spaces (default)
# split(“-”) : that means split the string into list using – dash ( if has character ).
# if put character to the function the function will split by the character.
# example:
a = 'I Love Python'
print(a.split())    # output is : [“I” ,”Love” , “python” ]

# example:
a = 'I-love-python'
print(a.split('-'))     # output is : [‘i’,’love’,’python’]
print(a.split('-',2))   # 2 means max split , make split to 2 elements and others in one
element .

# rsplit(): means split element in the right side of the items .


# 8- center()
# - that take the maximum number of characters , and the character
# [!,@,#,$,%,^,&,*] .
# - example :---
name = 'osama'
print(name.center(12,'#'))  # output is : ###osama###


# 9- count()
# - that count the characters of the specific element in the text.
# - you must write the character (s) to search about it in the string.
# - can determine the start and the end
# - example:
f = 'sdfghjkledfghjkrdfcvbn'
```

```
print(f.count('h')) # count the number the specific character .
print(f.count('d' ,0 ,14)) # 0 is the start element , 14 is the end element .


# 10-swapcase()
# used to replace elements [capital to small , small to capital ]
# example :---
name = 'osama'
print(name.swapcase()) # output is : OSAMA

# example :---
name = 'OSAMA'
print(name.swapcase()) # output is : osama


# 11-startwith()
# check the word is start with specific character ? , and it return Boolean value .
# need 3 components (at least one component ”1”) :
# 1- the word .
# 2- start character .
# 3- end character .
name = 'youssef'
print(name.startswith('y')) # output is : True .


# 12-endwith()
# check the word is end with specific character ? , and it return Boolean value .
# need three components (at least one component ”1”)

myName = 'osama'
print(myName.endswith('a'))
```

---

15 strings methods part 3

```
# 13- index()
# - Used to search about substring in the text and return it’s index .
# - take three components (at least one component “1”) : ---
# 1- Substring .
# 2- Start .
# 3- End .
# - Example :----
name = 'osama mohamed'
print(name.index('m'))   # output is : 3

# Note That : if the substring is not exist it will through an error .
# - Example : ----
a = 'abcdefghij'
print(a.index('a')) # output is : 0


# 14-Find()
# - Used to search about substring in the text and return it’s index .
# - Example :
```

```python
a = 'abcdefghij'
print(a.find('f'))   # output is : 5

# - Note That :---
# If the substring is not exist the output will be “ -1 “


# 15- rjust() ljust()
# rjust : means right justify .
# ljust : means left justify .
# take : width , the special character .
# o If don’t determine the character it will write spaces instead of the
# character
# example : ---
name = 'osama'
print(name.rjust(10,'*'))   # output is : *****osama


# 16-splitlines()
# that return all lines in list .
# example :
e = '''
first line
second line
third line
'''
print(e.splitlines())


# 17- expandtabs()
# that control number of spaces of the tab .
# Note That : tab consists of 8 spaces .
a = 'hello\tworld\tthis is first app'
print(a.expandtabs(2)) # means num of spaces in tab = 2

# =====================
# === quick methods ===
# =====================

# ❖ istitle() : means the text is title or not , return Boolean value .
x1 = "Osama"
print(x1.istitle())

# ❖ isspace() : means the text is space or not .
sp = " "
print(sp.isspace())

# ❖ islower() : means the text is lower case or not .
name = 'osama'
print(name.islower())

# ❖ isupper() : means the text is upper case or not .
name = 'OSAMA'
print(name.isupper())

# ❖ isidentifier() : means the text is suitable for to be a variable or not .
ident = "Osama"
```

```
print(ident.isidentifier())

# ❖ isalpha() : means the text is alphabetical character or not (a-z , A-Z).
text = 'osama mohamed'
print(text.isalpha())

# ❖ isalnum() : means the text is alphabet with numbers or not ( a-z , A-Z , 1-9 ) .
al = "this is text with numbers"
print(al.isalnum())
```

---

16 strings methods part 2

```
# 18- replace(old value ,new value ,count)
# - that used to replace string with string , you can determine num of strings that will
# replace .
# - take old value, new value , and number of strings that will replaced in the test .
a = 'Hello One Two Three One One'
print(a.replace("One" , '1'))
# Note That : new value should be string .


# 19- join() .
# join(iterable)
# - This convert list to string .
# - this function determine the separator before use it . convert the list , tuple into
string .
# Note That : Iterable Object can be tuple , list , …
# example :
myList = ['osama', 'ahmed', 'sayed', 'omar']
print(' ,'.join(myList))
```

---

17 string formatting old method

```
name = 'osama'
age = 36
rank = 10
print('name is : '+name +" and age is : "+age)  # type error

print("name is : %s " %name) # %s : means place holder

# That Note : الحاجه اللي بعد % هي القيمه بتاعه holder place
# another example : ---

name = ' Osama Mohamed '
age = 21
rank = 10
print('name Is : %s' %name)
print('name is : %s %d %d' %(name , age , rank))

# Note That : --
# - %s : means place holder .
# - %s : means that the place holder is string
```

```
# - %d : means that the place holder is digits
# - %f : means that the place holder is floating point number
# ❖ You can determine the number after the point number :--
number = 10
print('my number is %.2f : '%number)    # %.2f : means that we need 2 point number

# after the point .
# ❖ truncate (slice) the string : ----
# o you can determine the number of the string to print it :---
# Example :---
myLongString = 'hello peoples of el zero web school'
print('message is : %.5s' %myLongString)
# %.5s : means that the number of the letters is 5
```

---

18 string formatting new method

```
# Examples :----
# - case of string , int , …
print('my name is : {}'.format('osama'))
print('my age is : {}'.format( 22 ))

# - case of variables
name = 'osama'
print( 'my name is : {}'.format( name ) )

# ➔ Note That : ----
# :s that means variable is a string .
# :d that means variable is a integer .
# :f that means variable is a float .
# 1- control numbers after points :
# - you can control number of points in the floating point number
# - Example is :
Number = 10
print('my number is: {:.2f}'.format(Number))

# 2- Truncate the strings :
# - You can truncate the character (strings) .
# - Example : --
name = "osama"
print('my name is : {:.2s}'.format(name))


# 3- Format the numbers (money) : ---
# - You can format money .
# - Example :
myMoney = 23456789023
print('my money is : {:,}'.format(myMoney))
# will format the number after 3 numbers will put dot .
# ----> you can't determine all the character .
```

```
# 4- Rearranging items : ----
# - Note That : You can control the arrange by the element’s index .
a , b , c = 'one' , 'two' , 'three'
print('hello {} {} {}'.format(a , b , c))       # output : hello one two three
print('hello {1} {0} {2}'.format(a , b , c))    # output : hello two one three
x ,y ,z = 10 ,20 ,30
print('hello {2} {1} {0} '.format(a , b , c))
# output is : hello 30 20.00 30


# ==> Format in version 3.6+
name = 'osama'
age = 22
print(f"my name is {name} , and my age is {age}")
```

---

19 numbers

```
# → integer :--
print(type(11))
print(type(-11))
print(type(1097))

# → float :--
print(type(1.32))
print(type(-1.178))

# → complex number : ---
print(type(5+6j)) # that consists of two components (1- real numner , 2-imaginary )
myComplexNumber = 5+6j
print('number is : {}'.format(myComplexNumber))
print('real number is : {}'.format(myComplexNumber.real))
print('imaginary number is : {}'.format(myComplexNumber.imag))

# Some Notes : ---
# 1- You can convert from integer to float or complex .
# 2- You can convert from float to integer or complex .
# 3- You can’t convert complex to any type .
# ➔ Example : ---
print(100)
print(float(100)) # convert integer to float .
print(complex(100)) # convert integer to complex .
print(10.50)
print(int(10.50)) # convert float to integer .
print(complex(10.50)) # convert float to complex .
```

---

20 arithmetic operation

```
# [+] addition.
# [-] subtraction.
# [*] multiplication.
# [/] division.
# [%] modulus.
```

```
# [**] exponent .
# [//] floor division.

print(10+10)
print(10-2)
print(10/2)
print(10*2)
print(110//20)   # output : 5 → that the real number , that discard the point number .
print(12**2)     # means 12 power 2
```

---

21 list

```
# - List items are enclosed in square brackets .
# - List are ordered , use index to access elements .
# - List are mutable → add , delete , edit ,
# - List items isn't unique .
# - List can contain different data types .
# - You can use strings methods in list .
# - When use slicing the output is presented in list .

# ❖ Syntax :---
# List_name = ['element 1' , 1223 , True , False]

# ❖ Example :----
myList = [1,'hassan',True , False , 56 , 'osama']
print(myList[0])      # output is : 1
print(myList[2])      # output is : True
print([myList[-1]])   # output is : osama
print(myList[0:2])    # output is : [1, 'hassan']
print(myList[::1])    # output is : [1, 'hassan', True, False, 56, 'osama']
myList[0] = True      # edit the list
print(myList)         # output is : [True, 'hassan', True, False, 56, 'osama']
```

---

22 list methods part 1

```
# ❖ append() :---
# - used to insert new element to the end list .
# - append can be integer ,string ,Boolean ,float ,…
# - if you want to append list to list , the new list will be added as a list not
# added as elements .
# example :
myFriends = ['osama', 'mohamed', 'khaled']
myFriends.append('alaa')
print(myFriends)

# - we want to append list to list
list1 = [0,1,2,3,4,5,6]
```

```
list2 = ['zero', 'one', 'two', 'three', 'four', 'five']

list1.append(list2)     # append list1 to list2
print(list1)
print(list1[7][2:5])


# ❖ extend() :---
# - used to insert element to the list .
# - if you want to add list to list , the new list will added as elements .
# example :--
list1 = [0,1,2,3,4,5,6]
list2 = ['zero','one','two','three','four']
list1.extend(list2)
print(list1)


# remove() :---
# - used to remove element(s) by it's name .
# example :----
names = ['osama', 'mohamed', 'khaled']
names.remove('osama')
print(names)


# ❖ sort() :---
# - used to sort the numbers in the list .
# - used to sort the strings in the list .
# - can not sort string with integer .
num = [-1,23,55,0,2,4,10]
num.sort()
print(num)

# Note That :-- you can reverse the sorting
n = [0,234,56,24,5,778]
n.sort(reverse=False) # by default is false .
print(n)


# ❖ reverse() :---
# - used to only reverse the list
n = [1, 2, 3]
n.reverse()
print(n)
```

---

23 list methods part 2

```
# ❖ clear() :----
# - used to remove all items from list .
a = [1,2,3,4,5,6]
a.remove()
print(a)
```

```
# ❖ copy() :---
# - used to copy list’s element .
# example :---
a = [1,2,3]
b = a.copy()
print(a) # output is : [1,2,3]
print(a) # output is : [1,2,3]


# ❖ count() :---
# - used to count how many the specific element in the list .
# example :---
a = [1,2,3,5,2,3,41,1,1,3]
print(a.count(1))


# ❖ index() :----
# - used to search about “item” in the list ,and return it’s index.
# example : ----
names = ['osama', 'mohamed', 'khaled', 'ahmed']
print(names.index('osama')) # output is : 2

# ❖ insert() :---
# - used to insert object {element} before index
# - you should determine the element and it’s index .
# example : ---
a = [1,2,3,4,5,6,7,8]
a.insert(0, 'add to the first') # insert to the index 0
a.insert(8,'add to the end')    # insert to the index 8
print(a)


# ❖ pop() : ---
# - used to remove the last element (the default).
# - Used to remove a specific element in the list .
# - Example :---
N = [1,2,3,4,5]
N.pop()
print(N) # [1,2,3,4]
```

---

24 tuples

```
# 1- Tuples items are enclosed in parentheses () .
# 2- You can remove the parentheses if you want .
# 3- Tuple are ordered , to use index to access item .
# 4- Tuples are immutable => you can’t add or delete
# 5- Tuple items is not unique .
# 6- Tuples can have different data type .
# 7- Operators used in strings and lists available in tuples .
# Example :----
Tuple1 = (1,2,3,4,5)
Tuple2 = 1,2,34,45,6,87
Tuple3 = (True,12.12,17, 'osama', 'hassan')
```

---

25 tuples methods

```
# - How different between string and tuple if the value is one element ?
# By using comma after the value : this means that the type of it is a tuple
t = 'osama',    # type is Tuple
t = 'osama'     # type is string

# Tuple concatenation :---
a = [1,2,3,4]
b = [5,6]
c = a+b
print(c)
print(c+(11,223,123,True)) # another way of concatenation

# Tuple ,list , string repeat :---
string = 'osama'
myList = ['osmaa']
myTuple = ('osama')
print(string * 6) # repeat the string
print(myList * 6) # repeat the string
print(myTuple* 6) # repeat the string

# ➔ we repeat the string ,list ,tuple using *
# ? Tuple Methods :---
# ❖ Count() :-
# - Used to calculate how many the specific value are repeated .
# - Example :---
a = (1,2,3,4,2,1,3,54,2,1)
print(a.count(2))


# ❖ Index() :--
# - Used to print the index of a specific value .
# Example :---
a = (1,2,3,4,5,6,1)
print(a.index(3))
# - Another example :
a = (0,1,2,3,4,5,6)
print('the position of the value is : {:d}'.format(b.index(3)))

# ❖ tuple destruct :---
# example :--
a = ('A', "B", "C")
x,y,z = a
# example :---
a = ('A', "B", "C", "D")
x,y,z,_ = a

# Note That :- underscore used to ignore the value in the tuple.
```

26 sets

```
# 1- set items are enclosed in curly braces .
# 2- set items is not ordered ant not indexed .
# 3- set indexing and slicing can’t be done .
# 4- set has only immutable data types (numbers, strings, tuples) lists and
# dictionary are not .
# 5- set items is unique .
# - Note That : set items can’t be repeat , if repeat will ignore the repeat .
mySet = {} # syntax of the set .
mySet = {1,2,3,4,5}
print(mySet)
```

---

27 sets methods part 1

```
# 1- clear() : used to remove all element from the set .
a = {1,2,3,4,5}
print(a)

# 2- union() : that union more than one set .
b = {'one', 'two', 'three'}
c = {1,2,3}
print(b.union(c)) # using union function .
print(b|c) # or using “ | “ means union syntax .


# 3- add() : used to add new element to the set . that take one argument
# not more than one .
a = {1,2,3}
a.add(1) # true .
a.add(123 , 32) # false .

# 4- copy() : that copy element to new set .
a = {1,2,34,5}
b = a.copy()
print(b) # output is : {1,2,34,5}


# 5- remove() : used to remove a specific element from the set , if the specific
# element not exist on the set will case error .
a = {1,2,4,5}
a.remove(1)
print(a)


# 6- discard() : used to remove element from the set , if the element is not exist
# that will not case an error .
a = {1,2,4,5}
a.discard(1)
print(a)
```

```
# 7- pop() : used to remove random element from the set .
a = {1,2,34,45}
a.pop()
print(a)


# 8- update() : used to update a set with the union of itself and others
a = {1,2,3,4}
b = {'t', 'c', 1, 1, 2}
a.update(b)
print(a)
```

---

28 sets methods part 2

```
# 1- difference() : used to differ between two sets , that print the elements that
# exist on the first set and not exist on the second set .
# example :---
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
print(a.difference(b))
print(b)


# 2- difference_update() : used to get the difference then update the original
# set with the new elements .
# example :--
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
a.difference_update(b)
print(b)


# 3- intersection() # print the elements that exist on the first set and the
# second set .
# example :---
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
print(a.intersection(b))
print(b)

# 4- intersection_update() # print the elements that exist on the first set and
# the second set then update the original set with the new element .
# example:--
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
a.intersection_update()
print(b)
```

```
# 5- symmetric_difference() # get the elements the not exist on the first set
# and the second set .
# example :---
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
print(a.symmetric_difference(b))
print(b)


# 6- symmetric_difference_update () # get the elements the not
# exist on the first set and the second set . and update the original set .
# example :---
a = {1, 2, 3, 4, 5, 6, 'x'}
b = {'osama', 'zero', 1, 2, 4}
print(a)
a.symmetric_difference_update(b)
print(b)
```

---

29 sets methods part 3

```
# This methods return Boolean value (true , false)
# 1- issuperset() # check if the second set exist on the first set .
# example :--
a = {1,2,3,4}
b = {1,2,3}
print(a)
print(a.issuperset(b))


# 2- issubset() # check if the elements in the first set is exist on the second set ?
# example :--
a = {1,2,3,4}
b = {1,2,3}
print(a)
print(a.issubset(b))


# 3- isdisjoint() # check if the elements on both sets is disjoint (first set
# contain elements that doesn't exist on the second set )
# - return true if the first set contain elements that doesn't exist on the
# second set .
# - return false if the first set contain elements that exist on the second set .
# example :-
a = {1,2,3}
b = {4,5,6}
print(a.isdisjoint(b)) # true
# another example :
a = {1,2,3}
b = {1,2,3}
print(a.isdisjoint(b))
```

---

30 Dictionary

```
# o Dict items are enclosed in curly braces .
# o Dict items contains key : value .
# o Dict keys need to be immutable → (numbers, string ,tuple) list not allowed
# .
# o Dict value can have any data types .
# o Dict keys need to be unique .
# o Dict is not ordered , you access it’s element with keys .
# Example :--
user = {
    'name' : 'osama' ,
    'age' : 36 ,
    'country' : 'egypt',
    'skills' : ['html, css, js'],
    'rating' : 10.5 ,
    'name' : 'ahmed' # dict keys need to be unique
}
print(user)
print(user.keys()) # can print all keys of the dict .
print(user.values()) # print all values of the dict .
print(user.items()) # print the key and the value .
print(user.get('name')) # get used to get the specific key .


# ➔ two dimensional dictionary .
# Also called nested dictionary .
# Example : ---
languages = {
    'one' : {
    "name" : "html",
    "progress" : "80%"
    },
    'two' : {
    "name" : "CSS" ,
    "progress" : "90%"
    },
    "three" : {
    "three" : "js",
    "progress" : "70%"
    }
}
print(languages) # print all keys & values
print(languages["one"]) # print first dict on this .(this is first key)
# the following get key that exist on a subdict dictionary (dict exist on dict)
print(languages["two"]["name"])
print(len(languages)) # number of element that exist on the dict .
print(len(languages["two"])) # number of element that exist on subdict dict .
# Note That : I Mean about subdict is a dict that exist on other dict .


framework1 = {
    "name" : "Vuejs",
```

```
    'progress' : "90%"
}
framework2 = {
    "name" : "Reactjs",
    "progress" : "88%"
}
framework3 = {
    "name" : "angular" ,
    "progress" : "87%"
}
# I want to make dict to get the 3 frameworks in one dict .
allFrameworks = {
    "one" : framework1,
    "two" : framework2,
    "three" : framework3
}

# print all dicts , this collect all variables "dictionaries" in one dictionary
print(allFrameworks)
```

---

31 dictionary methods

```
# 1- Clear() # used to clear all items on the dictionary .
# Example :--
user = {
    "name" : "osama",
    'grade' : "A+",
    'age' : 21
}
print(user.clear())
print(user)


# 2- Update() # used to update item to the dict
# Example :---
member = {
    'name' : "osama"
}
member["age"] = 36 # used to append to the dictionary
print(member)
member.update( {"country" :"egypt"} )
print(member)


# 3- Copy() # used to copy dictionary items to another dictionary
# Example :--
a = {"name":"osama"}
b = a.copy()
print(b)
# Note That : the operations that made after copy not affect to the copied dictionary
```

```
# 4- Key() + value()
# Example:---
a = {'name':'osama',"age" : 35}
print(a.keys())
print(a.values())
```

---

32 dictionary methods part 2

```
# 1- setdefault() :
# used to search about the key that written in the method .
# if the key exist that will get the value of the key .
# if not exist that will write the value that written on the method .
# if the key is exist .
print('='*77)
user = {'name' : "osama"}
print(user)
print(user.setdefault("name","osama"))
print(user)

# if the key is not exist .
a = {"age" : 21}
print(a)
print(a.setdefault("name" , "osama"))
print(a)
# Note That : if you did not write the value of the key , python will write it as
None .
# None : it’s a data type that means nothing .


# 2- popitem() : this remove last element that exist on the dictionary .
a = {"age" : 21 ,
"name" : "osama"
}
print('='*66)
print(a)
print(a.popitem()) # delete the last item from the dictionary.
print(a)


# 3- items() : used to return all items that exist on the dictionary .if change any
# key , value this will change on the other variable .
# Note That : -- this method return items on list and every key and value in tuple .
a = {
"age" : 23,
"name" : "osama"
}
all = a.items()
print(a)
print(all)
```

```python
# 4- fromkeys() : make dict from variable and iterable .
a = {"key1" , "key2" , "key3"}
b = "X"
# a : refers to the keys
# b : refers to the values
print(dict.fromkeys(a,b))
```

---

33 Boolean Data Type

```python
# - in programming you need to know your if your code is True or False .
# - bool values are two constants objects : True or False .
name = ''
print(name.isspace()) # false .

# some examples
# true values
print(bool("osama"))
print(bool(120))
print(bool(-10))
print(bool(10.23))
# false values
print(bool("")) # empty value is false
print(bool('')) # empty value is false
print(bool([])) # empty value is false s
print(bool(0))
print(bool(None)) # null value (nothing value)
print(bool(False))
```

---

34 Boolean Operators

```python
# use it if we have more than one conditions
# - and : all condition are true
# example :----
name = "osama"
country = "egypt"
if name == "osama" and country == "egypt":
    print("hello osama in egypt")
# - or : one of the conditions are true .

# example :--
name = 'osama'
country = "egypt"
if name == "osama" or country == "cairo" :
    print('hello osama')
```

```
# - not : reverse the logical state .
age = 36
print(not age > 16)
```

---

35 Assignment Operators

```
# Assign : means assign a value to a variable .
# 1- = : variable is equal the value .
num1 = 10
num2 = 20
res = num1
print(res)


# 2- += : add first variable's value to the second variable's value
Means : var1 = var1 + var2
num1 = 10
num2 = 20
print(num1) # value of num1 = 10
num1 += num2 # value of num1 = 30 (10 + 20)
print(num1)


# 3- -= : subtract first value from the second value .
# Means : var1 = var1 – var2
num1 = 10
num2 = 5
num1 -= num2
print(num1) # 5


# 4- *= : multiply first value to the second value .
# Means : var1 = var1 * var2
num1 = 10
num2 = 2
num1 *= num2
print(num1) # 20


# 5- /= : divide first value to the second value .
# Means var1 = var1 / var2
num1 = 10
num2 = 2
num1 /= num2
print(num1)


# 6- **= :
# Means : var1 = var1 ** var2
num1 = 5
```

```
num2 = 2
num1 **= num2
print(num1)


# 7- %= :
# Means var1 = var1 % var2
num1 = 5
num2 = 2
num1 %= num2
print(num1)


# 8- //= :
# Means var1 = var1 // var2
num1 = 5
num2 = 2
num1 //= num2 # that get the real number .
print(num1)
```

---

36 Comparison Operators

```
# ✅ [ == ] called equal .
# ✅ [ != ] called not equal .
# ✅ [ > ] called greater than .
# ✅ [ < ] called less than .
# ✅ [ >= ] called greater than or equal .
# ✅ [ <= ] called less than or equal .
# Examples :---

print(100 == 100) # True
print(100 == 200) # False
print(100 == 100.00) # True
print(100 != 100) # False
print(100 != 200) # True
print(100 != 100.00) # False
print(100 > 100) # False
print(100 < 100) # False
print(100 > 200) # False
print(100 < 200) # True
print(100 <= 100) # True
print(100 >= 100) # True
print(100 <= 200) # True
print(100 >= 200) # False
```

---

37 Type Conversion

```
# 1- str() : used to convert int , float ,… to string
a = 10
print(type(str(a)))
print(type(a))


# 2- tuple() : used to convert string , list , dict , set to tuple .
# can not convert not iterable to tuple .
c = "osama" # strings
d = [1,2,3,4,5] # list
e = {"a", "b", "c"} #set
f = {"A" : 1, "B" : 2} # dictionary
print(tuple(c))
print(tuple(d))
print(tuple(e))
print(tuple(f))


# 3- list() : convert string , tuple , set , dict to list
c = "osama" # strings
d = (1,2,3,4,5) # tuple
e = {"a", "b", "c"} #set
f = {"A" : 1, "B" : 2} # dictionary
print(list(c))
print(list(d))
print(list(e))
print(list(f))
# Note That : in dictionary while converting the key is only convert .


# 4- set() : used to convert string, list, tuple, to set .
# Note That : Set elements not arranging .
a = "osama"
b = ('osama', 'mohamed', 'hassan')
c = ['osama', 'mohamed', 'hassan']
d = {"one" : 1, "two" : 2 }
print(set(a))
print(set(b))
print(set(c))
print(set(d))


# 5- dict() :
# ✅ can convert nested tuple to dictionary .
a = (("A",1) ,("B",2) , ("C",3))
print(dict(a))
# ✅ can convert nested list to dictionary .
a = [["a",1] , ["b",2] ,["c",3]]
print(dict(a))
# ❌ can not convert string to dictionary .
# ❌ can not convert set to dictionary .
# set is unhashable .
```

```
# ⍰ can not convert tuple to dictionary only convert nested tuple .
# ⍰ can not convert list to dictionary only convert nested list .
```

---

38 User Input

```
input() # is used to get input from the user .

fname = input('enter your first name : ')
mname = input('enter your middle name : ')
lname = input('enter your last name : ')
print( f"hello : {fname} {mname} {lname} happy to see you .")

# can use any string method :---
name = input('enter your name : ')
print(f"hello : {name.strip()}")
print(f"hello : {name.title()}")
print(f"hello : {name.capitalize()}")

# Note That : can write more than one method beside each other .
name = input('enter your name : ')
print(f"hello : {name.strip().title()}")

# Note That : you can truncate from the string , integer ,. ..
fname = input('enter your first name : ')
mname = input('enter your middle name : ')
lname = input('enter your last name : ')
print(f"hello : {fname.strip().title()} {mname.strip().title():.1s}
{lname.strip().title()}")

# Note That to the following (chain the methods) : ---
fname = input('enter your first name : ')
mname = input('enter your middle name : ')
lname = input('enter your last name : ')
fname = fname.strip().title()
mname = mname.title().strip()
lname = lname.title().strip()
print(f"hello : {fname} {mname:.1s} {lname}")
```

---

39 practical slice Email

```
# Get the position of the specific character, text, number, …
email = "Osama@gmail.org"
print(email.index('@'))

# Get the text before @ using the following method :-
email = "Osama@gmail.org"
print(email[0:email.index('@')])

# full example :--
name = input('enter your name : ')
```

```
email = input('enter your email : ')
print(f"name is : {name} ,and email is : {email}")
username = email[:email.index("@")]
domain = email[email.index("@")+1:]
print(f"the user name is : {username} ,and the domain is : {domain}")
```

---

40 practical your age details

```
age = int(input('what \'s your age ? ')) # take age from user as integer .

# get age in all time units .
months = age * 12
weeks = months * 4
days = age * 365
hours = days * 24
minutes = hours * 60
seconds = minutes * 60
print("You lived for : ")
print(f"{months} Months.")
print(f"{weeks} Weeks.")
print(f"{days:,} Days.")
print(f"{hours:,} Hours.")
print(f"{minutes:,} Minutes.")
print(f"{seconds:,} Seconds.")
```

---

41 control flow

```
# control flow
# if, elif, else
# make decisions
# syntax :--
# if condition:
#     statement
# elif condition:
#     statement
# else:
#     statement
# Example : -
country = input("enter your country : ")
if country == "egypt":
    print(f"the weather in {country} is : 15 ")
elif country == "KSA":
    print(f"the weather in {country} is : 20 ")
else :
    print("Country is not in the list .")
```

---

42 nested if

```
# Means if condition inside other if condition .
# Example :--

country = input("what \'s your country ? ")
age = int(input("enter your age : "))
if country == "egypt":
    if age == 33:
        print("wellcome from giza")
    elif age == 36:
        print("wellcome from cairo")
    else:
        ("wellcome")
elif country == "KSA":
    if age == 44:
        print("el salam alikom.")
else:
    print("Hello")
```

---

43 ternary conditional operators

```
# also called short if .
# ====================================
# === ternary conditional operator ===
# ====================================

movieRate = 18
age = 18
print("Movie is not good for you" if age < movieRate else "movie is good for you")

# === syntax ===
# ==============
# condition if true | if condition | else | condition if false
```

---

44 calculating age advanced version

```
# The following are the old method: ---
age = 21
months = age * 12
weeks = age * 4
days = age * 365
hours = days * 24
minutes = hours * 60
seconds = minutes * 60
print(f"you lived for : {months}\n")
```

```python
# the following is the new method : --
# ====================================
# == calculating age advanced version ==
# ====================================

# collect age Data :
age = int(input("enter your age : ").strip())

# Note Message
print("="*77) # string repeat
print("You can write the first letter or full name of the time unit ".center(77,"="))
print("="*77) # string repeat

# collect time unit Data :
unit = input("please choose time unit : months, weeks, Days .\n").strip().lower()

# get time unit
months = int(age) * 12
weeks = months * 4
days = int(age) * 365

if unit == "months" or unit == "m" :
    print("You Choose The Unit Months .")
    print(f"You Lived For {months:,} Months")
elif unit == "weeks" or unit == "w" :
    print("You Choose The Unit Weeks .")
    print(f"You Lived For {weeks:,} Weeks")
elif unit == "days" or unit == "d" :
    print("You Choose The Unit Days .")
    print(f"You Lived For {days:,} days")
```

---

45 Membership Operators

```python
# ============================
# === membership operators ===
# ============================
# Note That : string is case sensetive .
# in
# not in ➔ this reverse the logic

name = "osama"
print("s" in name) # true
print('='*55) # sperator
friends = ["osama","khaled","omar"]
print("Osama" in friends)
print('='*55) # sperator

# using in and not in with condition
countriesOne = ["Egypt", "KSA", "Kuwaite", "Bahrain"]
countriesOneDiscount = 80

countriesTwo = ["Italy", "Roma", "USA", "Russia"]
```

```
countriesTwoDiscount = 50

myCountry = input("enter your country : ")
if myCountry in countriesOne :
    print(f"you have a discount equal : {countriesOneDiscount} $ .")
else :
    print("you haven't discount !!")
```

---

46 practical membership control

```
# list contains admins
admins = ["Ahmed", "Osama", "Sameh", "Manal", "Rahma", "Mahmoud", "Enas"]

# login
name = input("Enter Your Name : ").strip().capitalize()

# check if the name in the admin list r not ?
if name in admins :
    print(f"Hello {name}, wellcome back .")

# this option to delete or update your name ?
    option = input("Delete or Update Your Name ? ").strip().capitalize()

    # update option
    if option == "Update" or "U" :

        # take new name from the admin
        theNewName = input("What \'s Your New Name ? ").strip().capitalize()

        # this change the old name that the admin will change it
        admins[admins.index(name)] = theNewName
        print("Name Updated .")

    # delete option
    elif option == "Delete" or "D":
        admins.remove(name)
        print("Name Deleted .")

    # if the user write another name
    # wrong option
    else :
        print("Wrong Option !!")

else :
    # ask user if add him admin or no ?
    status = input("Not Admin, Add You or No ?").strip().capitalize()

    # Yes Option, if the user want to add himself
    if status == "Yes" or "Y" :
        print('You Have Been Added .')
        # the name that the user entered at the first .
        admins.append(name)
```

```
    # No Option, if the user didn't want to add himself
    else :
        print("You Are Not Added ! ")


# print all names in the admins list
for n in admins:
    print(n.title())
```

---

47 While Loop

```
# while cindition_is_true
# code will run

# the code will run with infinte loop
# the run will stop if the condtion not achieve .
# a = 0
# while a < 10 :
# print("Hello World \n")

b = 0
while b < 10 :

    # if the condition true .
    print(f"the number is : {b} .")
    b += 1

# if the contion become fales .
else :
    print("Loop Is Done .")

# the will not print anything because the conditon is false
while False :
    print("will not print")
```

---

48 While Loop Training

```
# while cindition_is_true
# code will run

myF = ["osama", 'omar', 'khaled','ahmed','mohamed', 'gamal','hassan']
a = 0
while a < len(myF):

    # first "a" refers to the numbering of the names
    # second "a" refers to the element inside the list
    print(f"{a+1} : {myF[a].capitalize()}")
    a += 1 # this is the increment, and without it will make infinite loop
```

---

49 Simple Bookmark Manage

```
# create empty list to fill later
myFavoriteWebs = []

# maximux allowed websites
maximumWebs = 5

while maximumWebs > 0 :

    # input the new website name
    web = input("enter the name of the website without https:// : \n")

    # add the new website to the list
    myFavoriteWebs.append(f"https://{web.strip().lower()}")

    # decrease number from the max allowed websites
    maximumWebs -= 1

    # print the add website
    print(f"Website Added, {maximumWebs} places exist .")

# If No Exist Any Places
else :
    print("Bookmark is full, You Can't Add More . ")

    # check if list is not empty
    if len(myFavoriteWebs) > 0 :

        # sort the list
        myFavoriteWebs.sort()

        # loop in it
        index = 0
        print("printing the list of websites .")

    while index < len(myFavoriteWebs) :

        # print the list elements
        print(myFavoriteWebs[index])

        # the increment
        index += 1
```

50 While Loop Training Password Guess

```
# number of tries to write the password
tries = 4

# my paswword
mainPasswoed = "osama"

# this the password that the user input .
inputPassword = input("write your password : \n")

# if the password not true
while inputPassword != mainPasswoed :
    # this is the decrement
    tries -= 1

    # we use the short if
    # short if ==> if tries is the last try to write the password
    print(f"Wrong Password, and the number of tries is : {'Last' if tries == 0 else
tries} chances .")

    # try again to write the password
    inputPassword = input("write your password : \n")

    # check if the number of tries is finished break the program
    if tries == 0 :
        # print the following message if the tries finished
        print("you have used all your tries, try again nearly .")

    # and any command after "break" will not print
    break

# if the password true
else :
    print("successfully .")
```

---

51 For Loop

```
# for item in iteratable_object:
# do something with item
# item is a variable you created and call whenever you want
# item: refers to the current position and will run and visit all items to the end
# iteratable_object: sequence [list, tuple, set, dict, string, ...]
name = 'osama'
for letter in name:
    print(letter)

numbers = [0, 1, 2, 3, 4]
for number in numbers:
    print(number)
```

---

52 practical for loop

```
myRange = range(1, 100)
for number in myRange:
    print(number)


mySkills = {
    "HTML" : '90',
    "CSS" : '22',
    "JS" : '55',
    "PHP" : '44',
    "MySQL" : '77',
}
# print the value of the specific key
print(mySkills['JS'])
# print the value of the specific key
print(mySkills.get('PHP'))

# print the key and value
for skill in mySkills:
    print(f'language is {skill}, value is {mySkills.get(skill)}')
```

---

53 nested for loop

```
people = ['osama', 'mohamed', 'khaled']
skills = ['html', 'css', 'js']

for name in people: # outer loop

    print(f'{name.title()} Skills is: ')
for skill in skills: # inner loop
    print(f"- {skill}")

peoples = {
    'osama' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    }, 'osama' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    },
    'ahmed' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    },
    'mohamed' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    }
}
```

```python
# print the value of the specific dictionary
print(peoples['osama'])

# used to get the value that exist inside another value
print(peoples['osama']['css'])

# loop in dictionary
for name in peoples:
    print(f'==> {name} ')
    for skill in peoples[name]:
        print(f'- {skill}')
```

---

54 Break, Continue, Pass

```python
# continue: stop current iteration, then complete after it
# break: used to stop the loop if condition true
# pass: used to pass the loop
numbers = [1,2,3,4,6,7]
for num in numbers:
    if num == 4:
        continue
    print(num) # will print number without 4

print('='*44)

for num in numbers:
    if num == 4:
        break
    print(num) # will stop at 4
for num in numbers:
    pass
```

---

55 Loop Advanced

```python
peoples = {
    'osama' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    }, 'osama' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    },
    'ahmed' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    },
```

```
    'mohamed' : {
    'html' : '90%',
    'css' : '90%',
    'js' : '90%',
    }
}
for name, value in peoples.items(): # this is the main key
    print(f'Name Is : {name.title()}')
    for child_name, child_value in value.items(): # this is the inner key and its
value
        print(f"- {child_name.upper()} ==> {child_value}")
```

---

56 Function and Return

```
# a function is a reusable block of code that do a task
# a function run when you call ita function accept element to deal with called
[parameters]
# a function can do the task without returing data
# a function can return data after job is finished
# a function create to prevent DRY [don't repeat yoursel]
# a function accept elements when you call it calles argument
# there is a built-in functions and user defined functions\
# a function is for all team and all Apps
# syntax of the function
def function_name():
    print('hello world')

# function call
function_name()
```

---

57 Function and Parameter

```
def say(name):
    print(f'hello {name}')
# function call
say('osama')
# def: function keywork [define function]
# say(): function name
# name: parameter
# print(f"hello {name}"): this is the task
# say("ahmed"): ahmed is the argument
# say('osama'): function call

# example
def add(n1, n2):
    if type(n1) == int or type(n2) == int:
        print(n1+n2)
    else:
        print("Only Integer Allowed")
```

```
# function call
add(10, 20)
```

---

58 Packing, Unpacking Function Argument

```
myList = [1,2,3,4]
# *name: used to unpack the argument
print(*myList) # output: 1 2 3 4

# unpacking argument
def say_hello(*people):
    for name in people:
        print(f"Hello {name.title()}")

# function call
say_hello('osama')
say_hello('osama', 'khaled')
say_hello('osama', 'khaled', 'ahmed')
```

---

59 Function Default Parameter

```
# this used to put default parameter in the function
# to avoid the error, if the user don't want to write the argument

def say_hello(name, age, country=" "):
    print(f"name is {name}, and age is {age}, and the country is {country}")

# function call
say_hello('osama', 12)
```

---

60 Packing, Unpacking Arguments **kwargs

```
def show(*skills):
    for skill in skills:
        print(f"{skill.upper()}")

# Note That: the output is "tuple"
# function call
show('html', 'css')
def show(**skills):
    for skill, value in skills.items():
        print(f"the skill is: {skill}, and the value is: {value}")

# Note That: the output is "Dict"
# function call
show(HTML = '90%', CSS = "98%")
```

---

61 packing, unpacking training

```python
mySkills = {
    "HTML" : "80%",
    "CSS" : "80%",
    "JS" : "80%",
    "PHP" : "80%"
}

def show_skills(name, *skills, **skillsWithProgress):
    print(f"Hello {name.title()}")

    # unpacking the tuple [first loop]
    for skill in skills:
        print(f"- {skill}")

    # unpacking the dict [second loop]
    print(f"Skills with progress is: ")
    for key, value in skillsWithProgress.items():
        print(f"{key}, and the value is: {value}")
        # example of first loop
    show_skills('osama', 'HTML', "CSS", "JS")
    print("="*77)

# example to the second loop
show_skills('osama', 'HTML', **mySkills)
print("="*77)

# *skills: are tuple
# **skills: are Dict
my_dict = {
    "HTML" : "80%",
    "CSS" : "80%",
    "JS" : "80%",
    "PHP" : "80%"
}
my_tuple = ('HTML', "CSS", "JS")

def unpacking_tuple(*tuple):
    print(f"the skills is: ")
    for skill in my_tuple:
        print(f"- {skill}")

# function call
unpacking_tuple(*my_tuple)
print("="*77)

# function call
def unpacking_dict(**dict):
    print("the skills is: ")
    for key, value in my_dict.items():
        print("{key} {value}")
```

```
# function call
unpacking_dict(**my_dict)
```

---

62 function scope

```
x = 1 # this is global scope
print(f"print variable from global variable: {x }")

# function can get the variable from the global scope
def one():
    print(f'print variable from local scope: {x}')

# function call
one()
def two():
    x = 2
    print(f'print variable from local scope: {x}')
    # each function get its value from its place

# function call
two()

def three():
    # global used to make the variable global variable
    global x
    x = 3
    print(f'the variable from the local scope, but convert it to global: {x}')

# function call
three()
```

---

63 Function Recursion

```
def cleanWord(word):
    # if the length of the word = 1, return this word
    if len(word) == 1 :
        return word

    # if index[0] == index[1]
    if word[0] == word[1] :
        # return the function with index[1] to end
        return cleanWord(word[1:])

    # put the index[0], and concatenate with the function (the function start with
index[1])
    return word[0] + cleanWord(word[1:])

# function call
print(cleanWord("wwwooooorrldd"))
```

---

64 Lambda Function

```
# [1] it has no name
# [2] you can call it inline without defining it
# [3] you can use it in return data from another function
# [4] lambda used for simple functions and def handle the large tasks
# [5] lambda is one single expression not block of code
# [6] lambda type is function

# if the fuction single code, you can write it in single line
def say_hello(name) : return f"Hello {name.title()}"

# function call
print(say_hello('osama'))
# ================================================

# example of lambda function
# assign the function to variable
hello = lambda name : f"Hello {name.title()}"

# lambda function call
print(hello("osama"))

# ===============
# == Note That ==
# ===============
# lambda function has no name
# lambda: define lambda function
# name: this is parameter
# f"Hello {name.title()}": this is the single code that will execute
print(type(hello)) # function
```

---

65 File Handling

```
# "a" Append open file for appending values, create file if not exist
# "r" Read [default value] open file for read, case error if the file is not exist
# "w" Write open file for writing, create file if not exist
# "x" create create file, give error if the file exist
# open(): this take the file name with the path, and mode
file = open("C:\\Users\YOUSSEF\Desktop\osama.txt")
# close the file
file.close()
# ===============
# == Note That ==
# ===============
# there are two types of paths
# 1- absolute path: this start with the root to the specific file
# 2- relative path: this related with the your area [working directory]

import os
# get the main current working directory
print(os.getcwd())
```

```python
# get the working directory of the file
print(os.path.abspath(__file__))

# print the directory of the opened file
print(os.path.dirname(os.path.abspath(__file__)))

# change the current working directory, in this file [not globally]
os.chdir(os.path.dirname(os.path.abspath(__file__)))
# ==========================================

# ===============
# == Note That ==
# ===============
# if the path contains any escape sequence charatcter, use "r"
# "r": used to make the path raw string
file = open(r"D:\\Python\Files\nfile\osama.txt")
```

---

66 read files

```python
# open file for read
file = open("C:\\Users\YOUSSEF\Desktop\osama.txt", "r")

# print the "Data Object" of the file
print(file)

print(file.name) # print the path of the file

print(file.mode) # print the mode of the file

print(file.encoding) # print the encoding of the file

# read(): used to read the content of the file
# the default value is [-1], this read all the content of the file
# read all the content of the file
print(file.read())

# you can read specific characters from the file [limit read]
print(file.read(4)) # this read only 4 bytes from the file

# readline(): this used to read line from the file
print(file.readline())

# readlines(): used to read all lines from the file
# the output as list
print(file.readlines())
# read the file using for loop
for line in file:
    print(line)

    if line.startswith('4'):
        break

# close the file
file.close()
```

---

67 write & append to the file

```
# ➔Write to the file

# open file for read
# if the file not exist will create new file
# write remove the old content, and write the new content
file = open(r"C:\Users\YOUSSEF\Desktop\osama.txt", 'w')

# writet to the file
file.write("Hello Osama\n")
file.write("Hello Osama\n")
file.write("Hello Osama\n")

# writelines(): need list to write them to the file
myList = ['osama\n', 'ahmed\n', 'ahmed\n']
file.writelines(myList)

# close the file
file.close()

# ➔Append to the file

# open file for append
file = open(r'C:Users\YOUSSEF\Desktop\osama.txt', 'a')
# ============
# == append ==
# ============
# append used to append to the file
# this write to the content of the file
# the append write the content at the "cursor position"
# this append to the file
file.write('osama')
# close the file
file.close()
```

---

68 Important info

```
# open file for append
file = open(r"C:\Users\YOUSSEF\Desktop\osama.txt", 'a')

# truncate(): used to truncate from the file, and remove the others
file.truncate(5)

# tell(): used to print the position of the cursor
# the new line in windows = 2-bytes
print(file.tell())

# close the file
file.close()
# ================================================

# open file for read
file = open(r'C:\Users\YOUSSEF\Desktop\osama.txt', 'r')
```

```
# seek(): used to put the cursor at the specific position
file.seek(2)
print(file.read())

# close the file
file.close()
# ============================================================

import os
# remove file
os.remove(r"C:\Users\YOUSSEF\Desktop\osama.txt")
```

---

69 Built-in Function part 1

```
# all(): take iterable
x = [1, 2, 3, 4]
# if all iteratables true
if all(x):
    print("all elements is true")

# if one of the iteratables is false
else:
    print("one of these elements is false.")
# ================================================

x = [1, 2, 3, 4, "false"]
# any(): if any iteratables if
if any(x):
# if one iteratable if true
    print('one of these elements is fales')

# if there is no elements is true
else:
    print('all elements is fales')
# ================================================

x = [1, 2, 3, 4, "false"]

# bin(): used to convert the number to binary
print(bin(100))
# id(): used to get the id of the object from memory
```

---

70 Built-in Functions Part 2

```
# sum(iterable, start)
# used to sum elements in the iterable
# start: this added to iterable
numbers = [1,2,3,4,5]
print(sum(numbers)) # 15
print(sum(numbers, 20)) # 35
# ============================================================
```

```python
# round(num, num of digits): this get the near number
print(round(150.455)) # 150
print(round(150.455, 2)) # 150.46
# ============================================================

# range(start, end, step)
# if no start will start from 0
# end must determined, end not included
# is no step will put step = 1
print(list(range(0))) # []
print(list(range(10))) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(0, 10, 2))) # [0, 2, 4, 6, 8]
# ============================================================

# print()
# the default separator is "space"
print('osama mohamed') # osama mohamed

# this include two message
print('osama', 'mohamed') # osama mohamed

# this include three message
print('osama', 'mohamed', 'khaled') # osama mohamed khaled

# change the seperator
print('osama', 'mohamed', 'khaled', sep=',') # osama,mohamed,khaled

# end of the print is new line "\n"
print("first line") # first line
print("second line") # second line

# change the end of the print
print("osama mohamed", end=" ")
print("osama mohamed", end=" ")
```

---

71 Built-in functions part 3

```python
# abs(): this get the absolute value of the number
# the distance btween your number and zero
# this convert the number to positive
num = -10
print(abs(num)) # 10
# ================================================================

# pow(base, exp, mod) => power
print(pow(2, 3)) # 2*2*2 => 8
print(pow(2, 3, 2)) # 2*2*2 % 2 => 0
# ================================================================

# min(): used to get the minimun value
print(min(1,2)) # 1
print(min(2, -10)) # -10
print(min("ahmed", "osama", "khaled")) # ahmed
# ================================================================
```

```python
# max(): used to get the maximum value
print(max(1,2)) # 1
print(max(2, -10)) # -10
print(max("ahmed", "osama", "khaled")) # osama
# ====================================================================

# slice(start, end, step):
a = ['A', 'B', 'C', 'D', 'E', 'F']
print(a[:5]) # ['A', 'B', 'C', 'D', 'E']
# this must take stop [end]
print(a[slice(5)]) # ['A', 'B', 'C', 'D', 'E']
print(a[slice(2, 5)]) # ['C', 'D', 'E']
```

---

72 Built-in Functions part 4

```python
# [1] map take a function + iterator
# [2] map called map because it map the function on every element
# [3] the function can be pre-defined function or lambde function
# use map with pre-defined function
def formatText(text):
    return f"- {text.title()} -"

myText = ['OSama', "AHMED", "sAYed"]
myFormatedText = map(formatText, myText)
for name in myFormatedText:
    print(name)

# another way
for name in map(formatText, myText):
    print(name)

# use map with pre-defined function
myFriends = ['OSama', "AHMED", "sAYed"]
for name in map(lambda text: f"-> {text.title()}", myFriends):
    print(name)
```

---

73 Built-in Functions Part 5

```python
# [1] filter take a function + iterator
# [2] filter run a function on every element
# [3] the function can be pre-defined function or lambda function
# [4] filter out all elements for switch the function return True
# [5] the function need to return Boolean value
# example 1
def checkNumber(num):
    if num > 10:
        # Note That: filter return True Values
        return num
```

```python
myNumbers = [1, 19, 10, 20, 100, 5]
myResult = filter(checkNumber, myNumbers)

for number in myResult:
    print(number)

print("#"*77)
#================================================
def checkNum(n):
    if n == 0:
        # Note That: filter: don not print False values
        # return n [this is false condition]
        return True

Numbers = [10, 20, 0, 87, 0]
result = filter(checkNum, Numbers)
for num in result:
    print(num)
print("#"*77)
#================================================
def checkName(name):
    # this is condition
    # this return true if happen
    return name.startswith('O')

myText = ["Osama", "Ahmed", "Omar", "Sayed", "Osman"]
myReturnedData = filter(checkName, myText)

for person in myReturnedData:
    print(person)
#================================================
myNames = ["osama", 'Ahmed', 'amir', "Amir"]
myReturnedNames = filter(lambda name: name.startswith("A"), myNames)
for p in myReturnedNames:
    print(p)
```

---

74 Built-in Functions part 6

```python
# [1] reduce take a function + iterator
# [2] reduce run a function on first and second element and give result
# [ ] then run function on result and third element
# [ ] then run function on result and fourth element and so on
# [3] till one element is left and this is the result of the Reduce
# [4] the function can be pre-defined function or lambda function
# to use reduce() function, you should import the module [from functools import reduce]

from functools import reduce
def sumALl(num1, num2):
    return num1 + num2
numbers = [1,23,4,90,6,7,8]
result = reduce(sumALl, numbers)
print(result)
```

---

75 Built-in Functions part 7

```
# enumerate(): this add counter to the iterable while making loop
# enumerate(iterable, start=0)
mySkills = ['HTML', 'CSS', 'JS', 'PHP', 'MySQL']
mySkillsWithCounter = enumerate(mySkills)
for c, s in mySkillsWithCounter:
    print(f"{c} - {s}")
# ============
# == output ==
# 0 - HTML
# 1 - CSS
# 2 - JS
# 3 - PHP
# 4 - MySQL
# ============================================================
# help(): used to help you to get the manual of the functions
print(help(print))
# ============================================================
# reversed(): this reverse the iterable
numbers = [0, 1, 2, 3, 4]
# this will return the object
print(reversed(numbers))
for num in reversed(numbers):
    print(num)
```

---

76 Built-in Modules part 1

```
# [1] module is a file contains a set of functions
# [2] you can import module in your App to help you
# [3] you can import multiple module
# [4] you can create your own modules
# [5] modules saves your time

# import main module
import random

# print the module info
print(random)

# random(): used to print random floating number
print(f"print random floating number {random.random()}")

# dir(): this ptint all data in the object
# print all functions, parameters, ...
print(dir(random))

# import one or two functions from module
from random import randint, random

# randint(): this take range [two numbers]
# print random integer number
```

```
print(f"print random integer {random(10, 20)}")
```

---

77 create your module

```
import sys
# this print paths of the system of python
# search about the module in these paths
print(sys.path)

# append new path
sys.path.append(r"C:\Users\YOUSSEF\Desktop\mod1.py")

# Alias
import sys as s

# import your module
import mod1
print(dir(mod1))
```

---

78 install external package

```
# [1] module vs package
# [2] external package downloaded from internet
# [3] you can install packages with python package manager "pip"
# [4] pip install the packages ant its dependcies
# [5] module list "https://docs.python.org/3/py-modindex.html"
# [6] packages and modules directory "https://pypi.org/"
# [7] pip manual "https://pip.pypa.io/en/stable/reference/pip_install/"
# this check the version of the pip, or is exist or not
# pip --version
# this show all packages that you have
# pip list
# install external package
# pip install pytube
# this install more than one package
# pip install rembg pytube pygame
# you can install the package with the specific version
# pip install pygame=1.0.4 # specific version
# pip install pygame>=1.0.4 # bigger than version
# upgrade the package, module
# pip install pip --upgrade
# pip install --user pip --upgrade # this if case any error
# if case amy error press "ctrl + p" then write ">reload"

import pyfiglet
import termcolor
print(dir(figlet))
print(pyfiglet.figlet_format("OSAMA"))
```

---

79 date and time intro

```
import datetime

# to know the content of the package
print(dir(datetime))
print(dir(datetime.datetime))

print("#"*77)

# print the current date and time
print(datetime.datetime.now())

# print the current year
print(datetime.datetime.now().year)

# print the current month
print(datetime.datetime.now().month)

# print the current day
print(datetime.datetime.now().day)

print("#"*77)

# print start and end of date
print(datetime.datetime.min) # this is the start
print(datetime.datetime.max) # this is the end

print("#"*77)

# print the current time
print(datetime.datetime.now().time())

print("#"*77)

# print the current hour
print(datetime.datetime.now().time().hour)

print("#"*77)

# print the current minute
print(datetime.datetime.now().time().minute)

print("#"*77)

# print the current second
print(datetime.datetime.now().time().second)

print("#"*77)

# print start and end of time
print(datetime.time.min) # this is the start
print(datetime.time.max) # this is the end

print("#"*77)

# print specific date
# (year, month, day)
```

```
# (year, month, day, hour, minute, second, microsecond)
print(datetime.datetime(1982, 10, 25))

print("#"*77)

# this is the birthdate
myBirthDay = datetime.datetime(1982, 10, 25)
dateNow = datetime.datetime.now()

print(f"My Birthday is {myBirthDay}, and ", end="")
print(f"Date Now is {dateNow}")

# print the number of days, hours, minutes, microminutes
print(f'i lived for {dateNow - myBirthDay}')

# print the number of days, hours, minutes, microminutes
print(f'i lived for {(dateNow - myBirthDay).days} Days')
```

---

80 format the date and time

```
# this site include more info about the date and time characters
# https://strftime.org

import datetime

# this is the default date
myBirthday = datetime.datetime(1982, 10, 25)
print(myBirthday)

# format the time
# this is the month
print(myBirthday.strftime("%B"))
print(myBirthday.strftime("%b"))

# this is the Day
print(myBirthday.strftime("%A"))
print(myBirthday.strftime("%a"))

# ========================
# == determine the date ==
# ========================
# this is the day "%d"
# this is the month "%B"
# this is the year "%Y"
# print the day, month, year
print(myBirthday.strftime("%d %B %Y"))

# print the month, day, year
print(myBirthday.strftime("%B %d %Y"))

# print the day, month
print(myBirthday.strftime("%d - %b"))
```

---

81 iterable VS iterator

```
# ==============
# == iterable ==
# ==============
# [1] object contains data that can be iterated Upon
# [2] Examples (string, list, set, tuple, dictionary)
myString = "osama"
for letter in myString:
 print(letter)
print("="*55)
myList = ["osama", "khaled", "ahmed", "sayed"]
for name in myList:
 print(f"{name.title()}")
print("="*55)
# ============================================================
# ==============
# == iterator ==
# ==============
# [1] object used to iterate over iterable using next() method return 1 element at a tiem
# [2] you can generate iterator from iterable when using iter() method
# [3] for loop already calls iter() method on the iterable behind the scene
# [4] gives "StopIteration" if there is No exist element
# examples: int, float, bool
# all the following will case error
# these elements not iterable
#
muNumber = 10
for part in muNumber:
    print(part)

print("="*55)

muNumber = 10.907
for part in muNumber:
    print(part)

myBool = True
for b in myBool:
    print(b)

# convert the iterator to iterable by using iter()
myIterator = iter(myString)
print(next(myIterator))
print(next(myIterator))
print(next(myIterator))
print(next(myIterator))
print(next(myIterator))
```

---

82 Generator

```
# [1] generator is a function with "yield" keyword instead of "return"
# [2] it support iteration and return generator iterator by calling "yield"
# [3] generator function can have one or more "yield"
# [4] by using next() it resume from where it called "yield" not from begining
```

```python
# [5] when called, its not start automated, its only give you the control
# ------------------------------------------------------------------------------
def myGenerator():
    yield 1
    yield 2
    yield 3
    yield 4

print(myGenerator())

myGen = myGenerator()
print(next(myGen))
print(next(myGen))
print(next(myGen))
print(next(myGen))

for number in myGenerator():
    print(number)
```