Table of Contents

Week 1

003 – Syntax And Your First App

```
print("I love python")
```

```
I love python
```

```
print("i love programming")
```

```
i love programming
```

```
print(1);print(2)
```

```
1
2
```

```
if True:
    print(1)
```

```
1
```

004 – Comments

```
# this is a comment
# -----------
# hola
#------------
```

```
print("hi") #inline comment
```

```
hi
```

```
#print("ignore this code")
```

```
'''
 not multiline comment
'''
```

```
'\n not multiline comment\n'
```

006 – Some Data Types Overview

```
type(10)
# all data in python is object
```

```
int
```

```
type(10.1)
```

```
float
```

```
type("hello")
```

```
str
```

```
type([1,2,3])
```

```
list
```

```
type((1,2,3))
```

```
tuple
```

```
print(type({"one":1}))
```

```
<class 'dict'>
```

```
print(type(1==1))
```

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

007 – Variables Part One

```
# syntax => [variable name][assignment operator][value]
myVariable="my value"
print(myVariable)
```

```
my value
```

```
my_value = "value"
print(my_value)
```

```
value
```

```
# print(name)
# name ="yay" will generate error
# must assign value first before printing
```

```
name = "baka" #single word
myName="baka"#camelCase
my_name="baka"  #snake_case
```

008 – Variables Part Two

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 Commands
Interpreted : Code Translated On The Fly During Executionm

```
x=10
x="hello"
print(x) #dynamically typed language
```

```
hello
```

```
help("keywords")#reserved words
```

```
Here is a list of the Python keywords.  Enter any keyword to get more
help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

```
a,b,c =1,2,3
print(a,b,c)
```

```
1 2 3
```

009 – Escape Sequences Characters

Escape Sequences Characters

\b => Back Space
\newline => Escape New Line + \ => Escape Back Slash
' => Escape Single Quotes
" => Escape Double Quotes
\n => Line Feed
\r => Carriage Return
\t => Horizontal Tab
\xhh => Character Hex Value

```
print("hello\bword")
```

```
hellword
```

```
print("hello\
python")
```

```
hellopython
```

```
print("i love \\python")
```

```
i love \python
```

```
print("i love sinqle quote \"")
```

```
i love sinqle quote "
```

```
print("hello \nworld")
```

```
hello
world
```

```
print("123456\rabcd")
```

```
abcd56
```

```
print("hello \tworld")
```

```
hello 	world
```

```
print("\x4F\x73")
```

```
Os
```

010 – Concatenation And Trainings

```
msg = "i love"
lang = "python"
print(msg + " "+lang)
```

```
i love python
```

```
full = msg + " "+ lang
print(full)
```

```
i love python
```

```
a = "first\
    second\
        third"
b = "A\
    B\
    C"
print(a)
print(b)
```

```
first    second        third
A    B    C
```

```
print(a + "\n"+b)
```

```
first    second        third
A    B    C
```

`print("hi"+1)` will produce an error

```
print(msg +" 1")
```

```
i love 1
```

Week 2

011 – Strings

```
myStringOne= "this is 'signle quote'"
print(myStringOne)
```

```
this is 'signle quote'
```

```
myStringTwo = "This is Double Quotes"
print(myStringTwo)
```

```
This is Double Quotes
```

```
myStringThree = 'This is Single Quote "Test"'
print(myStringThree)
```

```
This is Single Quote "Test"
```

```
myStringFour = "This is Double Quotes 'Test'"
print(myStringFour)
```

```
This is Double Quotes 'Test'
```

```
multi= """first
second
third
"""
print(multi)
```

```
first
second
third
```

```
test = '''
"First" second 'third'
'''
print(test)
```

```

"First" second 'third'
```

012 – Strings – Indexing and Slicing

[1] All Data in Python is Object
[2] Object Contain Elements
[3] Every Element Has Its Own Index
[4] Python Use Zero Based Indexing (Index Start From Zero)
[5] Use Square Brackets To Access Element
[6] Enable Accessing Parts Of Strings, Tuples or Lists

```
#index access single item
string= "i love python"
print(string[0])
print(string[-1])
print(string[9])
```

```
i
n
t
```

```
#slicing access multiple sequence items
#[start:end]
#[start:end:steps]
print(string[8:11])
print(string[:10])
print(string[5:10])
print(string[5])
print(string[:])
print(string[0::1])
print(string[::1])
print(string[::2])
```

```
yth
i love pyt
e pyt
e
i love python
i love python
i love python
ilv yhn
```

013 – Strings Methods Part 1

```
a = "  i love python   "
print(len(a))
print(a.strip())
print(a.rstrip())
print(a.lstrip())
```

```
18
i love python
  i love python
i love python
```

```
a = "    hi   "
print(a.strip())
```

```
hi
```

```
a ="#####hola#####"
print(a.strip("#"))
print(a.rstrip("#"))
print(a.lstrip("#"))
```

```
hola
#####hola
hola#####
```

```
b ="I Love 2d Graphics and 3g Technology and python"
print(b.title())
print(b.capitalize())
```

```
I Love 2D Graphics And 3G Technology And Python
I love 2d graphics and 3g technology and python
```

```python
c,d,e = "1","20","3"
print(c.zfill(3))
print(d.zfill(3))
print(e.zfill(3))
```

```
001
020
003
```

```python
g = "aHmed"
print(g.lower())
print(g.upper())
```

```
ahmed
AHMED
```

014 – Strings Methods Part 2

```python
a = "I love python"
print(a.split())
```

```
['I', 'love', 'python']
```

```python
a = "I-love-python"
print(a.split("-"))
```

```
['I', 'love', 'python']
```

```python
a = "I-love-python"
print(a.split("-",1))
```

```
['I', 'love-python']
```

```python
d = "I-love-python"
print(d.rsplit("-",1))
```

```
['I-love', 'python']
```

```python
e = "ahmed"
print(e.center(15))
print(e.center(15,"#"))
```

```
     ahmed
#####ahmed#####
```

```python
f = "I and me and ahmed"
print(f.count("and"))
print(f.count("and",0,10))# start and end
```

```
2
1
```

```
g = "I love Python"
print(g.swapcase())
```

```
i LOVE pYTHON
```

```
print(g.startswith("i"))
print(g.startswith("I"))
print(g.startswith("l",2,12))#start from second index
```

```
False
True
True
```

```
print(g.endswith("n"))
print(g.endswith("e",2,6))
```

```
True
True
```

015 – Strings Methods Part 3

```
a = "I Love Python"
print(a.index("P"))  # Index Number 7
print(a.index("P", 0, 10))  # Index Number 7
#print(a.index("P", 0, 5))  # Through Error
```

```
7
7
```

```
b = "I Love Python"
print(b.find("P"))  # Index Number 7
print(b.find("P", 0, 10))  # Index Number 7
print(b.find("P", 0, 5)) #-1
```

```
7
7
-1
```

```
c = "ahmed"
print(c.rjust(10))
print(c.rjust(10, "#"))
```

```
     ahmed
#####ahmed
```

```
d = "ahmed"
print(d.ljust(10))
print(d.ljust(10, "#"))
```

```
ahmed
ahmed#####
```

```
e = """First Line
Second Line
Third Line"""

print(e.splitlines())

['First Line', 'Second Line', 'Third Line']

f = "First Line\nSecond Line\nThird Line"

print(f.splitlines())

['First Line', 'Second Line', 'Third Line']

g = "Hello\tWorld\tI\tLove\tPython"
print(g.expandtabs(5))

Hello     World     I    Love Python

one = "I Love Python And 3G"
two = "I Love Python And 3g"
print(one.istitle())
print(two.istitle())

True
False

three = " "
four = ""
print(three.isspace())
print(four.isspace())

True
False

five = 'i love python'
six = 'I Love Python'
print(five.islower())
print(six.islower())

True
False

# to check if i can use a name as a variable
seven = "osama_elzero"
eight = "OsamaElzero100"
nine = "Osama--Elzero100"

print(seven.isidentifier())
print(eight.isidentifier())
print(nine.isidentifier())
```

```
True
True
False
```

```python
x = "AaaaaBbbbbb"
y = "AaaaaBbbbbb111"
print(x.isalpha())
print(y.isalpha())
```

```
True
False
```

```python
u = "AaaaaBbbbbb"
z = "AaaaaBbbbbb111"
print(u.isalnum())
print(z.isalnum())
```

```
True
True
```

016 – Strings Methods Part 4

```python
# replace(Old Value, New Value, Count)

a = "Hello One Two Three One One"
print(a.replace("One", "1"))
print(a.replace("One", "1", 1))
print(a.replace("One", "1", 2))
```

```
Hello 1 Two Three 1 1
Hello 1 Two Three One One
Hello 1 Two Three 1 One
```

```python
myList = ["Osama", "Mohamed", "Elsayed"]
print("-".join(myList))
print(" ".join(myList))
print(", ".join(myList))
print(type(", ".join(myList)))
```

```
Osama-Mohamed-Elsayed
Osama Mohamed Elsayed
Osama, Mohamed, Elsayed
<class 'str'>
```

017 – Strings Formatting Old Way

```python
name = "Osama"
age = 36
```

```
rank = 10
```

```
print("My Name is: " + name)
```

```
My Name is: Osama
```

```
print("My Name is: " + name + " and My Age is: " + age)
```

type error, cant concatenate string with int

```
print("My Name is: %s" % "Osama")
print("My Name is: %s" % name)
print("My Name is: %s and My Age is: %d" % (name, age))
print("My Name is: %s and My Age is: %d and My Rank is: %f" % (name,
age, rank))
```

```
My Name is: Osama
My Name is: Osama
My Name is: Osama and My Age is: 36
My Name is: Osama and My Age is: 36 and My Rank is: 10.000000
```

%s => String
%d => Number
%f => Float

```
n = "Osama"
l = "Python"
y = 10
```

```
print("My Name is %s Iam %s Developer With %d Years Exp" % (n, l, y))
```

```
My Name is Osama Iam Python Developer With 10 Years Exp
```

```
#control flow point number
myNumber = 10
print("My Number is: %d" % myNumber)
print("My Number is: %f" % myNumber)
print("My Number is: %.2f" % myNumber)
```

```
My Number is: 10
My Number is: 10.000000
My Number is: 10.00
```

```
#Truncate string
myLongString = "Hello Peoples of Elzero Web School I Love You All"
print("Message is %s" % myLongString)
print("Message is %.5s" % myLongString)
```

```
Message is Hello Peoples of Elzero Web School I Love You All
Message is Hello
```

018 – Strings Formatting New Way

```
name = "Osama"
age = 36
rank = 10

print("My Name is: " + name)
```

```
My Name is: Osama
```

```
print("My Name is: {}".format("Osama"))
print("My Name is: {}".format(name))
print("My Name is: {} My Age: {}".format(name, age))
print("My Name is: {:s} Age: {:d} & Rank is: {:f}".format(name, age,
rank))
```

```
My Name is: Osama
My Name is: Osama
My Name is: Osama My Age: 36
My Name is: Osama Age: 36 & Rank is: 10.000000
```

{:s} => String
{:d} => Number
{:f} => Float

```
n = "Osama"
l = "Python"
y = 10

print("My Name is {} Iam {} Developer With {:d} Years Exp".format(n,
l, y))
```

```
My Name is Osama Iam Python Developer With 10 Years Exp
```

```
# Control Floating Point Number
myNumber = 10
print("My Number is: {:d}".format(myNumber))
print("My Number is: {:f}".format(myNumber))
print("My Number is: {:.2f}".format(myNumber))
```

```
My Number is: 10
My Number is: 10.000000
My Number is: 10.00
```

```
# Truncate String
myLongString = "Hello Peoples of Elzero Web School I Love You All"
print("Message is {}".format(myLongString))
print("Message is {:.5s}".format(myLongString))
print("Message is {:.13s}".format(myLongString))
```

```
Message is Hello Peoples of Elzero Web School I Love You All
Message is Hello
Message is Hello Peoples
```

```
#format money
myMoney = 500162350198
print("My Money in Bank Is: {:d}".format(myMoney))
print("My Money in Bank Is: {:_d}".format(myMoney))
print("My Money in Bank Is: {:,d}".format(myMoney))
```

```
My Money in Bank Is: 500162350198
My Money in Bank Is: 500_162_350_198
My Money in Bank Is: 500,162,350,198
```

{:&d} will produce an error

```
# ReArrange Items
a, b, c = "One", "Two", "Three"
print("Hello {} {} {}".format(a, b, c))
print("Hello {1} {2} {0}".format(a, b, c))
print("Hello {2} {0} {1}".format(a, b, c))
```

```
Hello One Two Three
Hello Two Three One
Hello Three One Two
```

```
x, y, z = 10, 20, 30
print("Hello {} {} {}".format(x, y, z))
print("Hello {1:d} {2:d} {0:d}".format(x, y, z))
print("Hello {2:f} {0:f} {1:f}".format(x, y, z))
print("Hello {2:.2f} {0:.4f} {1:.5f}".format(x, y, z))
```

```
Hello 10 20 30
Hello 20 30 10
Hello 30.000000 10.000000 20.000000
Hello 30.00 10.0000 20.00000
```

```
# Format in Version 3.6+
myName = "Osama"
myAge = 36
print("My Name is : {myName} and My Age is : {myAge}")
print(f"My Name is : {myName} and My Age is : {myAge}")
```

```
My Name is : {myName} and My Age is : {myAge}
My Name is : Osama and My Age is : 36
```

Week 3

019 – Numbers

```
# Integer
print(type(1))
print(type(100))
print(type(10))
print(type(-10))
print(type(-110))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
<class 'int'>
```

```
# Float
print(type(1.500))
print(type(100.99))
print(type(-10.99))
print(type(0.99))
print(type(-0.99))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

```
# Complex
myComplexNumber = 5+6j
print(type(myComplexNumber))
print("Real Part Is: {}".format(myComplexNumber.real))
print("Imaginary Part Is: {}".format(myComplexNumber.imag))
```

```
<class 'complex'>
Real Part Is: 5.0
Imaginary Part Is: 6.0
```

[1] You Can Convert From Int To Float or Complex
[2] You Can Convert From Float To Int or Complex
[3] You Cannot Convert Complex To Any Type

```
print(100)
print(float(100))
print(complex(100))
```

```
100
100.0
(100+0j)
```

```
print(10.50)
print(int(10.50))
print(complex(10.50))
```

```
10.5
10
(10.5+0j)
```

```
print(10+9j)
#print(int(10+9j)) error
```

```
(10+9j)
```

020 – Arithmetic Operators

[+] Addition
[-] Subtraction
[*] Multiplication
[/] Division
[%] Modulus
[**] Exponent
[//] Floor Division

```
# Addition
print(10 + 30)
print(-10 + 20)
print(1 + 2.66)
print(1.2 + 1.2)
```

```
40
10
3.66
2.4
```

```
# Subtraction
print(60 - 30)
print(-30 - 20)
print(-30 - -20)
print(5.66 - 3.44)
```

```
30
-50
-10
2.22
```

```
# Multiplication
print(10 * 3)
print(5 + 10 * 100)
print((5 + 10) * 100)
```

```
30
1005
1500
```

```python
# Division
print(100 / 20)
print(int(100 / 20))
```

```
5.0
5
```

```python
# Modulus
print(8 % 2)
print(9 % 2)
print(20 % 5)
print(22 % 5)
```

```
0
1
0
2
```

```python
# Exponent
print(2 ** 5)
print(2 * 2 * 2 * 2 * 2)
print(5 ** 4)
print(5 * 5 * 5 * 5)
```

```
32
32
625
625
```

```python
# Floor Division
print(100 // 20)
print(119 // 20)
print(120 // 20)
print(140 // 20)
print(142 // 20)
```

```
5
5
6
7
7
```

021 – Lists

[1] List Items Are Enclosed in Square Brackets
[2] List Are Ordered, To Use Index To Access Item

[3] List Are Mutable => Add, Delete, Edit
[4] List Items Is Not Unique
[5] List Can Have Different Data Types

```
myAwesomeList = ["One", "Two", "One", 1, 100.5, True]

print(myAwesomeList)
print(myAwesomeList[1])
print(myAwesomeList[-1])
print(myAwesomeList[-3])
```

```
['One', 'Two', 'One', 1, 100.5, True]
Two
True
1
```

```
print(myAwesomeList[1:4])
print(myAwesomeList[:4])
print(myAwesomeList[1:])
```

```
['Two', 'One', 1]
['One', 'Two', 'One', 1]
['Two', 'One', 1, 100.5, True]
```

```
print(myAwesomeList[::1])
print(myAwesomeList[::2])
```

```
['One', 'Two', 'One', 1, 100.5, True]
['One', 'One', 100.5]
```

```
print(myAwesomeList)
myAwesomeList[1] = 2
myAwesomeList[-1] = False
print(myAwesomeList)
```

```
['One', 'Two', 'One', 1, 100.5, True]
['One', 2, 'One', 1, 100.5, False]
```

```
myAwesomeList[0:3]=[]
print(myAwesomeList)
```

```
[1, 100.5, False]
```

```
myAwesomeList[0:2] = ["A","B"]
print(myAwesomeList)
```

```
['A', 'B', False]
```

022 – Lists Methods Part 1

```python
myFriends = ["Osama", "Ahmed", "Sayed"]
myOldFriends = ["Haytham", "Samah", "Ali"]
print(myFriends)
print(myOldFriends)
```

```
['Osama', 'Ahmed', 'Sayed']
['Haytham', 'Samah', 'Ali']
```

```python
myFriends.append("Alaa")
myFriends.append(100)
myFriends.append(150.200)
myFriends.append(True)
print(myFriends)
```

```
['Osama', 'Ahmed', 'Sayed', 'Alaa', 100, 150.2, True]
```

```python
myFriends.append(myOldFriends)
print(myFriends)
```

```
['Osama', 'Ahmed', 'Sayed', 'Alaa', 100, 150.2, True, ['Haytham',
'Samah', 'Ali']]
```

```python
print(myFriends[2])
print(myFriends[6])
print(myFriends[7])
```

```
Sayed
True
['Haytham', 'Samah', 'Ali']
```

```python
print(myFriends[7][2])
```

```
Ali
```

```python
a = [1, 2, 3, 4]
b = ["A", "B", "C"]
print(a)
```

```
[1, 2, 3, 4]
```

```python
a.extend(b)
print(a)
```

```
[1, 2, 3, 4, 'A', 'B', 'C']
```

```python
x = [1, 2, 3, 4, 5, "Osama", True, "Osama", "Osama"]
x.remove("Osama")
print(x)
```

```
[1, 2, 3, 4, 5, True, 'Osama', 'Osama']
```

```python
y = [1, 2, 100, 120, -10, 17, 29]
y.sort()
print(y)
```

```
[-10, 1, 2, 17, 29, 100, 120]
```

```
y.sort(reverse=True)
print(y)
```

```
[120, 100, 29, 17, 2, 1, -10]
```

```
m = ["A", "Z", "C"]
m.sort()
print(m)
```

```
['A', 'C', 'Z']
```

Sort can't sort a list that contains both of strings and numbers.

```
z = [10, 1, 9, 80, 100, "Osama", 100]
z.reverse()
print(z)
```

```
[100, 'Osama', 100, 80, 9, 1, 10]
```

023 – Lists Methods Part 2

```
a = [1, 2, 3, 4]
a.clear()
print(a)
```

```
[]
```

```
b = [1, 2, 3, 4]
c = b.copy()

print(b)
print(c)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4]
```

```
b.append(5)
print(b)
print(c)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
```

```
d = [1, 2, 3, 4, 3, 9, 10, 1, 2, 1]
print(d.count(1))
```

```
3
```

```
e = ["Osama", "Ahmed", "Sayed", "Ramy", "Ahmed", "Ramy"]
print(e.index("Ramy"))
```

```
3
```

```
f = [1, 2, 3, 4, 5, "A", "B"]
print(f)
f.insert(0, "Test")
f.insert(-1, "Test")
print(f)
```

```
[1, 2, 3, 4, 5, 'A', 'B']
['Test', 1, 2, 3, 4, 5, 'A', 'Test', 'B']
```

```
g = [1, 2, 3, 4, 5, "A", "B"]
print(g.pop(-3))
```

```
5
```

024 – Tuples Methods Part 1

[1] Tuple 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] Tuple Are Immutable => You Cant Add or Delete
[5] Tuple Items Is Not Unique
[6] Tuple Can Have Different Data Types
[7] Operators Used in Strings and Lists Available In Tuples

```
myAwesomeTupleOne = ("Osama", "Ahmed")
myAwesomeTupleTwo = "Osama", "Ahmed"
print(myAwesomeTupleOne)
print(myAwesomeTupleTwo)
```

```
('Osama', 'Ahmed')
('Osama', 'Ahmed')
```

```
print(type(myAwesomeTupleOne))
print(type(myAwesomeTupleTwo))
```

```
<class 'tuple'>
<class 'tuple'>
```

```
myAwesomeTupleThree = (1, 2, 3, 4, 5)
print(myAwesomeTupleThree[0])
print(myAwesomeTupleThree[-1])
print(myAwesomeTupleThree[-3])
```

```
1
5
3
```

```python
# Tuple Assign Values
myAwesomeTupleFour = (1, 2, 3, 4, 5)
print(myAwesomeTupleFour)
```

```
(1, 2, 3, 4, 5)
```

```python
myAwesomeTupleFour[2] = "Three"
print(myAwesomeTupleFour)
```

'tuple' object does not support item assignment

```python
myAwesomeTupleFive = ("Osama", "Osama", 1, 2, 3, 100.5, True)
print(myAwesomeTupleFive[1])
print(myAwesomeTupleFive[-1])
```

```
Osama
True
```

025 – Tuples Methods Part 2

```python
myTuple1 = ("Osama",)
myTuple2 = "Osama",
print(myTuple1)
print(myTuple2)
```

```
('Osama',)
('Osama',)
```

```python
print(type(myTuple1))
print(type(myTuple2))
```

```
<class 'tuple'>
<class 'tuple'>
```

```python
print(len(myTuple1))
print(len(myTuple2))
```

```
1
1
```

```python
a = (1, 2, 3, 4)
b = (5, 6)
c = a + b
d = a + ("A", "B", True) + b
print(c)
print(d)
```

```
(1, 2, 3, 4, 5, 6)
(1, 2, 3, 4, 'A', 'B', True, 5, 6)
```

```
myString = "Osama"
myList = [1, 2]
myTuple = ("A", "B")

print(myString * 6)
print(myList * 6)
print(myTuple * 6)
```

```
OsamaOsamaOsamaOsamaOsamaOsama
[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
('A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B', 'A', 'B')
```

```
a = (1, 3, 7, 8, 2, 6, 5, 8)
print(a.count(8))
```

```
2
```

```
b = (1, 3, 7, 8, 2, 6, 5)
print("The Position of Index Is: {:d}".format(b.index(7)))
print(f"The Position of Index Is: {b.index(7)}")
```

```
The Position of Index Is: 2
The Position of Index Is: 2
```

```
# Tuple Destruct
a = ("A", "B", 4, "C")

x, y, _, z = a

print(x)
print(y)
print(z)
```

```
A
B
C
```

Week 4

026 – Set

[1] Set Items Are Enclosed in Curly Braces
[2] Set Items Are Not Ordered And Not Indexed
[3] Set Indexing and Slicing Cant Be Done
[4] Set Has Only Immutable Data Types (Numbers, Strings, Tuples) List and Dict Are Not
[5] Set Items Is Unique

```
# Not Ordered And Not Indexed

mySetOne = {"Osama", "Ahmed", 100}
print(mySetOne)
```

```
{100, 'Osama', 'Ahmed'}
```

`print(mySetOne[0])` will produce an error.

Slicing Cant Be Done

```
mySetTwo = {1, 2, 3, 4, 5, 6}
print(mySetTwo[0:3])
```

Will also produce an error.

Has Only Immutable Data Types

`mySetThree = {"Osama", 100, 100.5, True, [1, 2, 3]}`
Error, unhashable type: 'list'

```
mySetThree = {"Osama", 100, 100.5, True, (1, 2, 3)}

print(mySetThree)
```

```
{True, 100.5, 100, (1, 2, 3), 'Osama'}
```

```
# Items are Unique

mySetFour = {1, 2, "Osama", "One", "Osama", 1}
print(mySetFour)
```

```
{1, 2, 'Osama', 'One'}
```

027 – Set Methods Part 1

```
a = {1, 2, 3}
a.clear()
print(a)
```

```
set()
```

```
b = {"One", "Two", "Three"}
c = {"1", "2", "3"}
x = {"Zero", "Cool"}
print(b | c)
print(b.union(c))
print(b.union(c, x))
```

```
{'One', 'Two', '2', 'Three', '1', '3'}
{'One', 'Two', '2', 'Three', '1', '3'}
{'One', 'Two', 'Cool', 'Zero', '2', 'Three', '1', '3'}
```

```python
d = {1, 2, 3, 4}
d.add(5)
d.add(6)
print(d)
```

```
{1, 2, 3, 4, 5, 6}
```

```python
e = {1, 2, 3, 4}
f = e.copy()


print(e)
print(f)
e.add(6)
print(e)
print(f)
```

```
{1, 2, 3, 4}
{1, 2, 3, 4}
{1, 2, 3, 4, 6}
{1, 2, 3, 4}
```

```python
g = {1, 2, 3, 4}
g.remove(1)
# g.remove(7) will remove an error
print(g)
```

```
{2, 3, 4}
```

```python
h = {1, 2, 3, 4}
h.discard(1)
h.discard(7)# wont produce an error
print(h)
```

```
{2, 3, 4}
```

```python
i = {"A", True, 1, 2, 3, 4, 5}
print(i.pop())
```

```
True
```

```python
j = {1, 2, 3}
k = {1, "A", "B", 2}
j.update(['Html', "Css"])
j.update(k)

print(j)
```

```
{1, 2, 3, 'Html', 'A', 'Css', 'B'}
```

028 – Set Methods Part 2

```python
a = {1, 2, 3, 4}
b = {1, 2, 3, "Osama", "Ahmed"}
print(a)
print(a.difference(b))  # a - b
print(a)
```

```
{1, 2, 3, 4}
{4}
{1, 2, 3, 4}
```

```python
c = {1, 2, 3, 4}
d = {1, 2, "Osama", "Ahmed"}
print(c)
c.difference_update(d)  # c - d
print(c)
```

```
{1, 2, 3, 4}
{3, 4}
```

```python
e = {1, 2, 3, 4, "X", "Osama"}
f = {"Osama", "X", 2}
print(e)
print(e.intersection(f))  # e & f
print(e)
```

```
{1, 2, 3, 4, 'X', 'Osama'}
{'X', 2, 'Osama'}
{1, 2, 3, 4, 'X', 'Osama'}
```

```python
g = {1, 2, 3, 4, "X", "Osama"}
h = {"Osama", "X", 2}
print(g)
g.intersection_update(h)  # g & h
print(g)
```

```
{1, 2, 3, 4, 'X', 'Osama'}
{'X', 2, 'Osama'}
```

```python
i = {1,2,3,4,5}
j = {0,3,4,5}
print(i)
print(i.symmetric_difference(j))  # i ^ j
print(i)
```

```
{1, 2, 3, 4, 5}
{0, 1, 2}
{1, 2, 3, 4, 5}
```

```
i = {1,2,3,4,5}
j = {0,3,4,5}
print(i)
i.symmetric_difference_update(j)  # i ^ j
print(i)
```

```
{1, 2, 3, 4, 5}
{0, 1, 2}
```

029 – Set Methods Part 3

```
a = {1, 2, 3, 4}
b = {1, 2, 3}
c = {1, 2, 3, 4, 5}

print(a.issuperset(b))
print(a.issuperset(c))
```

```
True
False
```

```
d = {1, 2, 3, 4}
e = {1, 2, 3}
f = {1, 2, 3, 4, 5}

print(d.issubset(e))
print(d.issubset(f))
```

```
False
True
```

```
g = {1, 2, 3, 4}
h = {1, 2, 3}
i = {10, 11, 12}

print(g.isdisjoint(h))
print(g.isdisjoint(i))
```

```
False
True
```

030 – Dictionary

[1] Dict Items Are Enclosed in Curly Braces
[2] Dict Items Are Contains Key : Value
[3] Dict Key Need To Be Immutable => (Number, String, Tuple) List Not Allowed
[4] Dict Value Can Have Any Data Types
[5] Dict Key Need To Be Unique
[6] Dict Is Not Ordered You Access Its Element With Key

```
user = {
  "name": "Osama",
  "age": 36,
  "country": "Egypt",
  "skills": ["Html", "Css", "JS"],
  "rating": 10.5
}
print(user)
```

```
{'name': 'Osama', 'age': 36, 'country': 'Egypt', 'skills': ['Html',
'Css', 'JS'], 'rating': 10.5}
```

```
user = {
  "name": "Osama",
  "age": 36,
  "country": "Egypt",
  "skills": ["Html", "Css", "JS"],
  "rating": 10.5,
  "name": "Ahmed"
}
print(user)
```

```
{'name': 'Ahmed', 'age': 36, 'country': 'Egypt', 'skills': ['Html',
'Css', 'JS'], 'rating': 10.5}
```

- Notice that it prints Ahmed not Osama as it is defined later.

```
print(user['country'])
print(user.get("country"))
```

```
Egypt
Egypt
```

```
print(user.keys())
```

```
dict_keys(['name', 'age', 'country', 'skills', 'rating'])
```

```
print(user.values())
```

```
dict_values(['Ahmed', 36, 'Egypt', ['Html', 'Css', 'JS'], 10.5])
```

```
languages = {
  "One": {
    "name": "Html",
    "progress": "80%"
  },
  "Two": {
    "name": "Css",
    "progress": "90%"
  },
  "Three": {
    "name": "Js",
    "progress": "90%"
  }
}
```

```
print(languages)
```

```
{'One': {'name': 'Html', 'progress': '80%'}, 'Two': {'name': 'Css',
'progress': '90%'}, 'Three': {'name': 'Js', 'progress': '90%'}}
```

```
print(languages['One'])
```

```
{'name': 'Html', 'progress': '80%'}
```

```
print(languages['Three']['name'])
```

```
Js
```

```
print(len(languages))
```

```
3
```

```
print(len(languages["Two"]))
```

```
2
```

```
frameworkOne = {
  "name": "Vuejs",
  "progress": "80%"
}
frameworkTwo = {
  "name": "ReactJs",
  "progress": "80%"
}
frameworkThree = {
  "name": "Angular",
  "progress": "80%"
}
allFramework = {
  "one": frameworkOne,
  "two": frameworkTwo,
  "three": frameworkThree
}
print(allFramework)
```

```
{'one': {'name': 'Vuejs', 'progress': '80%'}, 'two': {'name':
'ReactJs', 'progress': '80%'}, 'three': {'name': 'Angular',
'progress': '80%'}}
```

031 – Dictionary Methods Part 1

```
user = {
  "name": "Osama"
}
print(user)
user.clear()
print(user)
```

```
{'name': 'Osama'}
{}
```

```
member = {
  "name": "Osama"
}
print(member)
member["age"] = 36
print(member)
member.update({"country": "Egypt"})
print(member)
# Both ways are equivalent.
```

```
{'name': 'Osama'}
{'name': 'Osama', 'age': 36}
{'name': 'Osama', 'age': 36, 'country': 'Egypt'}
```

```
main = {
  "name": "Osama"
}

b = main.copy()
print(b)
main.update({"skills": "Fighting"})
print(main)
print(b)
```

```
{'name': 'Osama'}
{'name': 'Osama', 'skills': 'Fighting'}
{'name': 'Osama'}
```

```
print(main.keys())
```

```
dict_keys(['name', 'skills'])
```

```
print(main.values())
```

```
dict_values(['Osama', 'Fighting'])
```

032 – Dictionary Methods Part 2

```
user = {
  "name": "Osama"
}
print(user)
user.setdefault("name","Ahmed")
user.setdefault("age", 36)
print(user)
```

```
{'name': 'Osama'}
{'name': 'Osama', 'age': 36}
```

```
print(user.setdefault("name","Ahmed"))
```

```
Osama
```

```
member = {
  "name": "Osama",
  "skill": "PS4"
}
print(member)
member.update({"age": 36})
print(member)
print(member.popitem())
print(member)
```

```
{'name': 'Osama', 'skill': 'PS4'}
{'name': 'Osama', 'skill': 'PS4', 'age': 36}
('age', 36)
{'name': 'Osama', 'skill': 'PS4'}
```

```
view = {
  "name": "Osama",
  "skill": "XBox"
}

allItems = view.items()
print(view)
```

```
{'name': 'Osama', 'skill': 'XBox'}
```

```
view["age"] = 36
print(view)
```

```
{'name': 'Osama', 'skill': 'XBox', 'age': 36}
```

```
print(allItems)
```

```
dict_items([('name', 'Osama'), ('skill', 'XBox'), ('age', 36)])
```

```
a = ('MyKeyOne', 'MyKeyTwo', 'MyKeyThree')
b = "X"

print(dict.fromkeys(a, b))
```

```
{'MyKeyOne': 'X', 'MyKeyTwo': 'X', 'MyKeyThree': 'X'}
```

```
user = {
  "name": "Ahmed"
}
me = user
print(me)
```

```
{'name': 'Ahmed'}
```

```
user["age"]=21
print(me)
```

```
{'name': 'Ahmed', 'age': 21}
```

- Notice that `me` got updated because `me` and `user` share the same data.

Week 5

033 – Boolean

[1] In Programming You Need to Known Your If Your Code Output is True Or False
[2] Boolean Values Are The Two Constant Objects False + True.

```
name = " "
print(name.isspace())
```

```
True
```

```
name = "Ahmed"
print(name.isspace())
```

```
False
```

```
print(100 > 200)
print(100 > 100)
print(100 > 90)
```

```
False
False
True
```

```
print(bool("Osama"))
print(bool(100))
print(bool(100.95))
print(bool(True))
print(bool([1, 2, 3, 4, 5]))
```

```
True
True
True
True
True
```

```
print(bool(0))
print(bool(""))
print(bool(''))
print(bool([]))
print(bool(False))
print(bool(()))
print(bool({}))
print(bool(None))
```

```
False
False
False
False
False
False
False
False
```

034 – Boolean Operators

```
name = "ahmed"
age = 21
print(name =="ahmed" and age == 21)
```

```
True
```

```
print(name =="ahmed" and age > 21)
```

```
False
```

```
print(name =="ahmed" or age > 21)
```

```
True
```

```
print(name =="mohamed" or age > 21)
```

```
False
```

```
print(not age > 21)
```

```
True
```

```
print(not (name =="ahmed" and age > 21))
```

```
True
```

035 – Assignment Operators

```
=
+=
-=
*=
/=
**=
%=
//=
```

```
x = 10
y = 20
x = x+y
print(x)
```

```
30
```

- A better way

```
a = 10
b = 20
a+= b
print(a)
```

```
30
```

```
x = 30
y = 20
x = x-y
print(x)
```

```
10
```

```
a = 30
b = 20
a -= b
print(a)
```

```
10
```

Var One = Self [Operator] Var Two
Var One [Operator]= Var Two

036 – Comparison Operators

[==] Equal
[!=] Not Equal
[>] Greater Than
[<] Less Than
[>=] Greater Than Or Equal
[<=] Less Than Or Equal

```
print(100 == 100)
print(100 == 200)
print(100 == 100.00)
```

```
True
False
True
```

```
print(100 != 100)
print(100 != 200)
print(100 != 100.00)
```

```
False
True
False
```

```
print(100 > 100)
print(100 > 200)
print(100 > 100.00)
print(100 > 40)
```

```
False
False
False
True
```

```
print(100 < 100)
print(100 < 200)
print(100 < 100.00)
print(100 < 40)
```

```
False
True
False
False
```

```
print(100 >= 100)
print(100 >= 200)
print(100 >= 100.00)
print(100 >= 40)
```

```
True
False
True
True
```

```python
print(100 <= 100)
print(100 <= 200)
print(100 <= 100.00)
print(100 <= 40)
```

```
True
True
True
False
```

037 – Type Conversion

```python
a = 10
print(type(a))
print(type(str(a)))
```

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

```python
c = "Osama"
d = [1, 2, 3, 4, 5]
e = {"A", "B", "C"}
f = {"A": 1, "B": 2}

print(tuple(c))
print(tuple(d))
print(tuple(e))
print(tuple(f))
```

```
('O', 's', 'a', 'm', 'a')
(1, 2, 3, 4, 5)
('A', 'B', 'C')
('A', 'B')
```

```python
c = "Osama"
d = (1, 2, 3, 4, 5)
e = {"A", "B", "C"}
f = {"A": 1, "B": 2}

print(list(c))
print(list(d))
print(list(e))
print(list(f))
```

```
['O', 's', 'a', 'm', 'a']
[1, 2, 3, 4, 5]
['A', 'B', 'C']
['A', 'B']
```

```
c = "Osama"
d = (1, 2, 3, 4, 5)
e = ["A", "B", "C"]
f = {"A": 1, "B": 2}

print(set(c))
print(set(d))
print(set(e))
print(set(f))
```

```
{'s', 'a', 'O', 'm'}
{1, 2, 3, 4, 5}
{'A', 'B', 'C'}
{'A', 'B'}
```

```
d = (("A", 1), ("B", 2), ("C", 3))
e = [["One", 1], ["Two", 2], ["Three", 3]]

print(dict(d))
print(dict(e))
```

```
{'A': 1, 'B': 2, 'C': 3}
{'One': 1, 'Two': 2, 'Three': 3}
```

038 – User Input

```
fName = input('What\'s Is Your First Name?')
mName = input('What\'s Is Your Middle Name?')
lName = input('What\'s Is Your Last Name?')

fName = fName.strip().capitalize()
mName = mName.strip().capitalize()
lName = lName.strip().capitalize()

print(f"Hello {fName} {mName:.1s} {lName} Happy To See You.")
```

```
Hello Ahmed H Darwish Happy To See You.
```

039 – Practical – Email Slice

```
email = "Osama@elzero.org"
print(email[:email.index("@")])
```

```
Osama

theName = input('What\'s Your Name ?').strip().capitalize()
theEmail = input('What\'s Your Email ?').strip()

theUsername = theEmail[:theEmail.index("@")]
theWebsite = theEmail[theEmail.index("@") + 1:]

print(f"Hello {theName} Your Email Is {theEmail}")
```

```
Hello Ahmed Your Email Is Ahmedh457@gmail.com
```

```
print(f"Your Username Is {theUsername} \nYour Website Is
{theWebsite}")
```

```
Your Username Is Ahmedh457
Your Website Is gmail.com
```

040 – Practical – Your Age In Full Details

```
age = int(input('What\'s Your Age ? ').strip())

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.")
```

```
You Lived For:
252 Months.
1,008 Weeks.
7,665 Days.
183,960 Hours.
11,037,600 Minutes.
662,256,000 Seconds.
```