

# PYTHON FOUNDATION

## MANUAL BOOK



Northern City

By

Tin Mai Zaw

(Published Year- 2024)

## အမှာစာ

ဤစာအုပ်သည် Python Programming ကို အခြေခံမှစတင် လေ့လာမည့် သူများအတွက် လက်စွဲစာအုပ်ဖြစ်ပါသည်။ ကွန်ပျူတာကျောင်း သူကျောင်းသားများ၊ အင်ဂျင်နီယာအိုင်တီ ကျောင်းသူကျောင်းသားများကို တစ်ဖက်တစ်လမ်း အထောက်အကူပြုစေရန်အတွက် ရည်ရွယ် ထုတ်ဝေရခြင်းလည်းဖြစ်ပါသည်။ အိုင်တီနဲ့ အသက်မွေးဝမ်းကြောင်းပြုလိုသူများ ၊ software programmer ဖြစ်ချင်သူများ ကမ္ဘာကျော် software application တွေတီထွင်ပြီး millionaire သူဌေးဖြစ်ဖို့ စိတ်ကူးအိမ်မက်ရှိသူ မည်သူမဆို လွယ်လွယ်ကူကူ လေ့လာနိုင်အောင် လေ့ကျင့်ခန်း ဥပမာများ ၊ သင်ခန်းစာ video tutorial များနဲ့ တွဲပြီးတော့ ပြည့်ပြည့်စုံစုံ ရေးသားဖန်တီးထားတဲ့ စာအုပ်ဖြစ်ပါသည်။ ဘွဲ့လွန် Master ၊ Phd တက်နေသော သူများ၊ ကိုယ်ပိုင် စာတမ်းပြုစုနေသူများ အတွက်လည်း data analysis foundation တွေကို လေ့လာနိုင်အောင် numpy ၊ pandas နဲ့ matplotlib လေ့ကျင့်ခန်းတွေကို အလွယ်ကူဆုံး နားလည်နိုင်အောင် ရေးသားဖော်ပြထားတဲ့ စာအုပ်တစ်အုပ်ဖြစ်ပါသည်။

## စာရေးသူ ကိုယ်ရေးအကျဉ်း

ဤစာအုပ်ကိုရေးသားပြုစုသူကတော့ ကျနော် ဆရာတင်မိုင်ဇော် ဖြစ်ပါတယ်။ ကျနော်က မြစ်ကြီးနားမြို့ အထက (၁) မှာ အထက်တန်းအောင်မြင်ခဲ့ပါတယ်။ ပြီးတော့ ကျနော်က ဖိလစ်ပိုင်နိုင်ငံ University of the Philippines ( Los Banos) မှာ B.Sc Computer Science ကိုဆက်လက်ပညာသင်ယူခဲ့ပါတယ်။ ၂၀၀၄ မှာ အိုင်တီနဲ့ ဘွဲ့ရကျောင်းပြီးခဲ့ပါတယ်။ ၂၀၀၅ မှာ မြန်မာပြည်ပြန်လာပြီး မြစ်ကြီးနား ဇာတိမြို့မှာ Northern City Computer Training Center ကွန်ပျူတာသင်တန်းကျောင်းကို စတင်တည်ထောင်ဖွင့်လှစ်ခဲ့ပါတယ်။ လေ့လာဆည်းပူးခဲ့တဲ့ programming IT ဘာသာရပ်တွေကို သင်ကြားပို့ချခဲ့တာ ဖြစ်ပါတယ်။ ၂၀၀၉ မှာ မလေးရှားနိုင်ငံ Kuala Lumpur မှာ zepto it solution လို့ခေါ်တဲ့ အိုင်တီ Company မှာ Software Developer ( programmer ) အဖြစ် ၃ နှစ်တာ အလုပ်လုပ်ခဲ့ပါတယ်။ ကျန်းမာရေးအခြေနေကြောင့် ရန်ကုန်မြို့ကို ပြန်လာပြီး ဆေးကုသရင်း ရန်ကုန်မြို့မှာပဲ ValueStar Computer Institute မှာ ၆ နှစ်တာ IT Lecturer အနေနဲ့ရော IT Department Head အနေနဲ့ရော ဆက်လက်ခြေချခဲ့ပါတယ်။ ၂၀၁၇ မှာ ကိုယ်ပိုင် အိုင်တီသင်တန်းကျောင်းအဖြစ် Northern City Center နာမည်နဲ့ အရင်က မြစ်ကြီးနားမှာ တည်ထောင်ခဲ့ဖူး တဲ့ နာမည်ကို ပြန်လည်အသုံးပြုပြီး ဖွင့်လှစ်တည်ထောင်ခဲ့တာဖြစ်ပါတယ်။

စာရေးသူ၏ အိုင်တီအတွေ့အကြုံ မှတ်တမ်းများ -

Popular Web projects –

- Aya Internet Banking  
<https://www.ayaibanking.com/>
- Japan Used Car Sales & Show room  
<https://www.sbtjapan.com/sbt-myanmar/>
- Myanmar Triditional Boxing  
<https://www.myanmartraditionalboxing.com.mm>
- Myanmar Agriculture Machinery & Products  
<https://tptyeeshinn.com.mm>
- Real Estate  
<https://www.saikhungnounge.com>
- Malaysia Minimart  
<https://familystore.com.my>
- China Products & Fashion Sales  
<https://youhome.space/>
- Online Payment System  
<https://ucapay.com.mm>

Popular Point of Sales Application Projects –

- Malaysia Family Store Desktop Application and Mobile Application
- Myitkyina iGu Café & Gallery Desktop Application
- Myitkyina Hospital Blood Bank Desktop Application
- Myitkina Fuji Food & Drinks Desktop Application
- Yangon Joyzone Stock Controller Desktop Application and Mobile Application
- Yangon Malihka Restaurant Desktop Application and Mobile Application
- Yangon Yuri Café Mobile Application and Mobile Application
- Yangon Gracevines Agarwood Desktop Application and Mobile Application

## Contents

---


1. Setting up Development Environment	Page 5
2. Datatypes	Page 14
3. Basic Calculations	Page 18
4. Variables & Comments	Page 23
5. Conditional Statements	Page 26
6. Loops	Page 32
7. Methods	Page 41
8. OOP	Page 50
9. Lists	Page 58
10. Tuples	Page 65
11. Dictionary	Page 74
12. File	Page 80
13. Datetime	Page 85
14. Math & Random	Page 88
15. Lambda	Page 92
16. Map	Page 95
17. Filter	Page 98
18. RegularExpression	Page 103
19. Numpy	Page 108
20. Pandas	Page 115
21. Matplotlib	Page 128

---

---

## *Setting up Development Environment*

---

 video tutorial 1

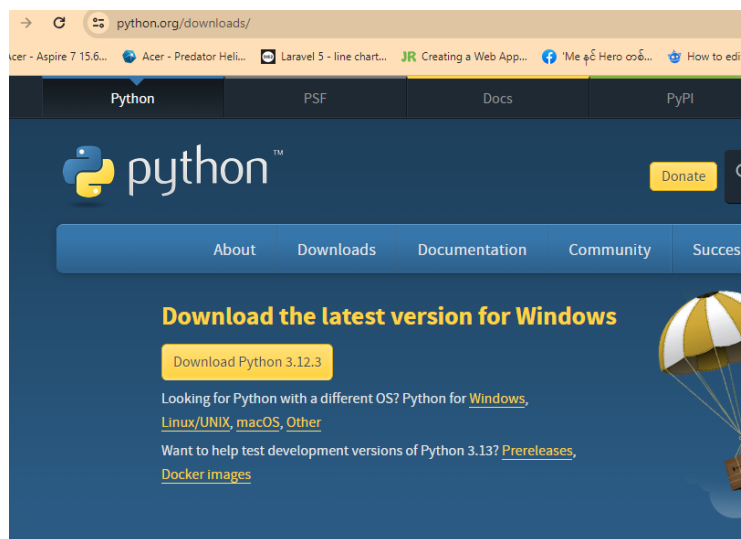


## Software Installation Steps –

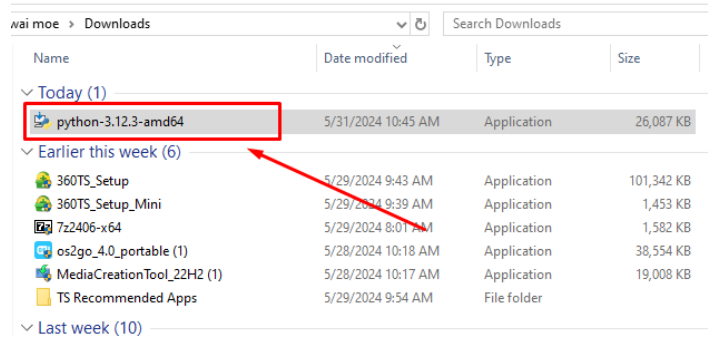
1. Python Installaion
2. PyCharm Installation
3. Running the first Python App








## Python Installation Guide

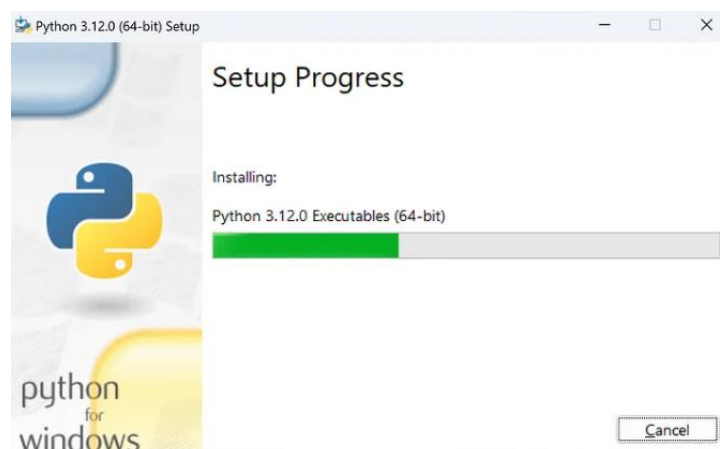
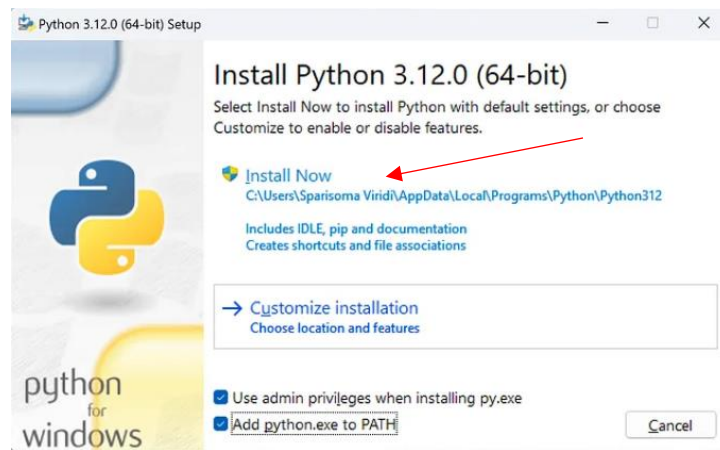
- Please download python SDK application from the following link  
<https://www.python.org/downloads/>

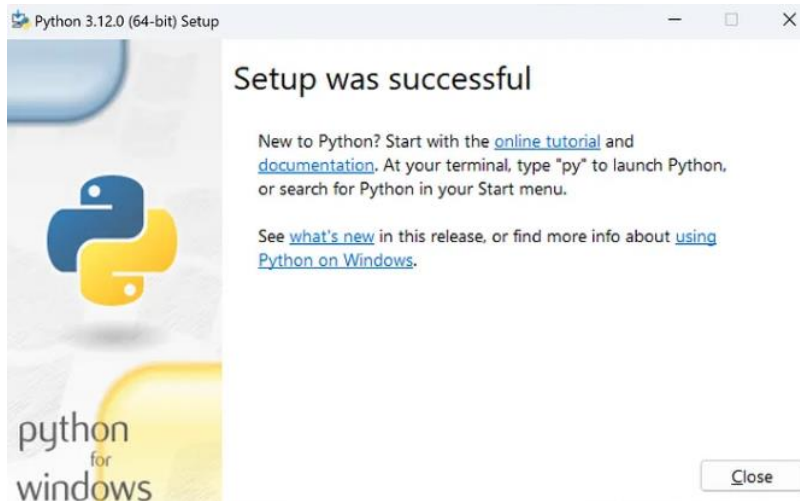


## Python Installation Process



Name	Date modified	Type	Size
▼ Today (1)			
 python-3.12.3-amd64	5/31/2024 10:45 AM	Application	26,087 KB
▼ Earlier this week (6)			
 360TS_Setup	5/29/2024 9:43 AM	Application	101,342 KB
 360TS_Setup_Mini	5/29/2024 9:39 AM	Application	1,453 KB
 7z2406-x64	5/29/2024 8:01 AM	Application	1,582 KB
 os2go_4.0_portable (1)	5/28/2024 10:18 AM	Application	38,554 KB
 MediaCreationTool_22H2 (1)	5/28/2024 10:17 AM	Application	19,008 KB
 TS Recommended Apps	5/29/2024 9:54 AM	File folder	
▼ Last week (10)			






Now you are ready to install pycharm

- Please go to the following link and download pycharm application
- Please choose community option

<https://www.jetbrains.com/pycharm/download/>



## Download PyCharm

Windows macOS Linux

### Professional

Full-featured IDE for Python & Web development

**DOWNLOAD**

Free trial

### Community

Lightweight IDE for Python & Scientific development

**DOWNLOAD**

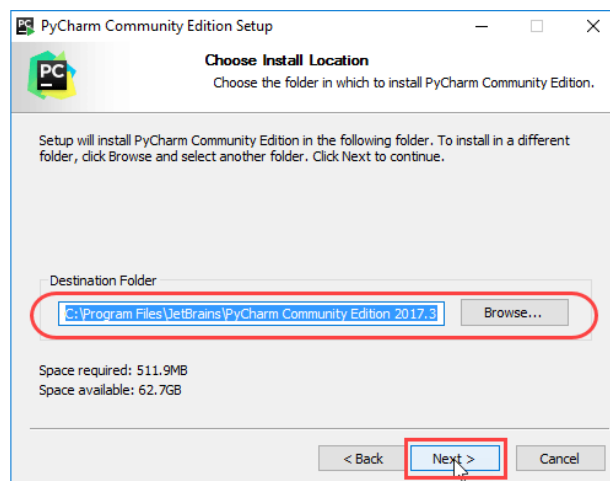
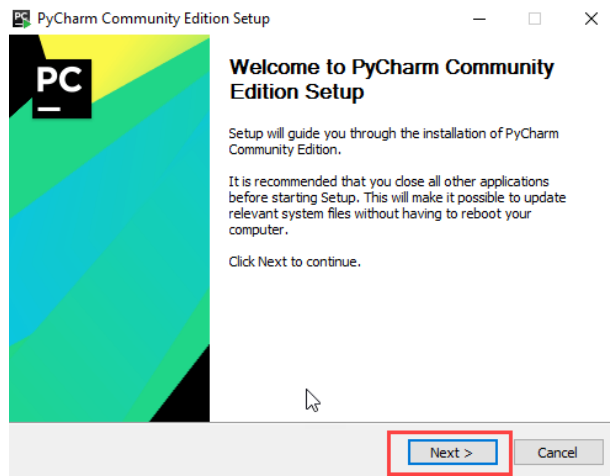
Free, open-source

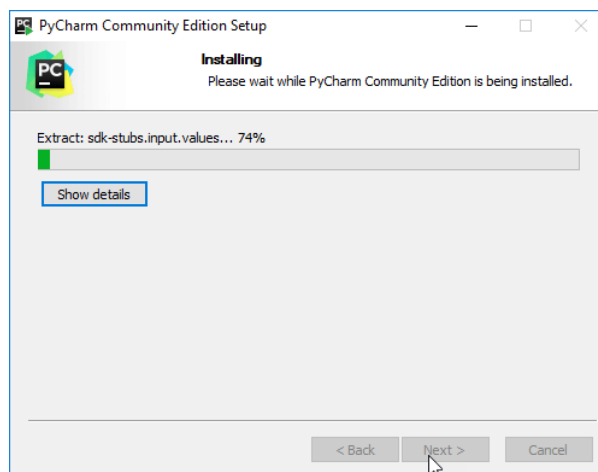
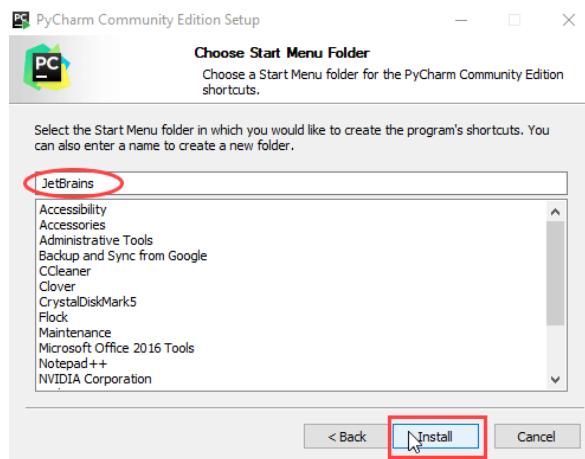
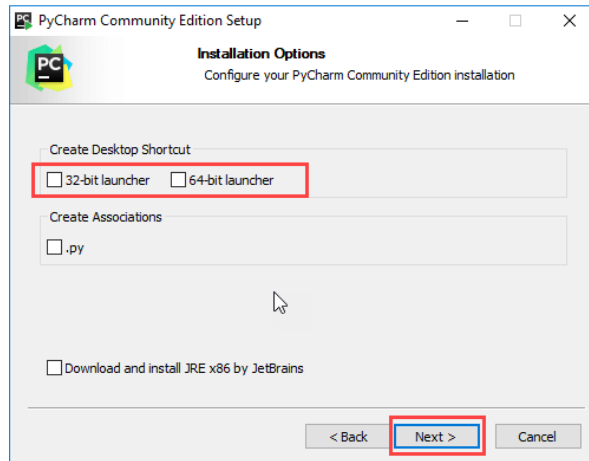
Version: 2017.2.2  
Build: 172.3757.67  
Released: August 24, 2017

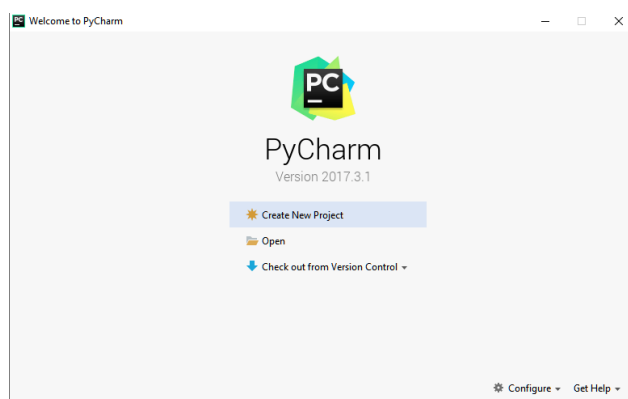
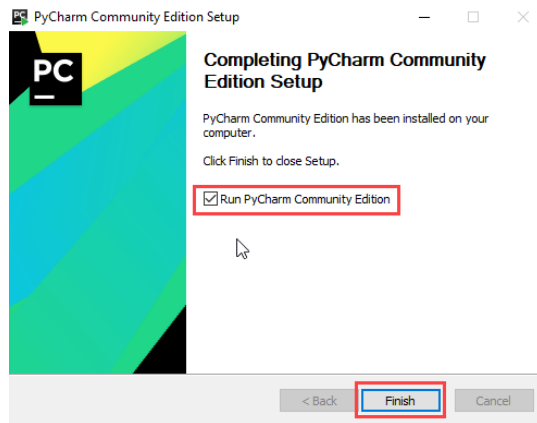
[System requirements](#)  
[Installation Instructions](#)  
[Previous versions](#)



After downloading pycharm community application, you have to install pycharm app as the followings:

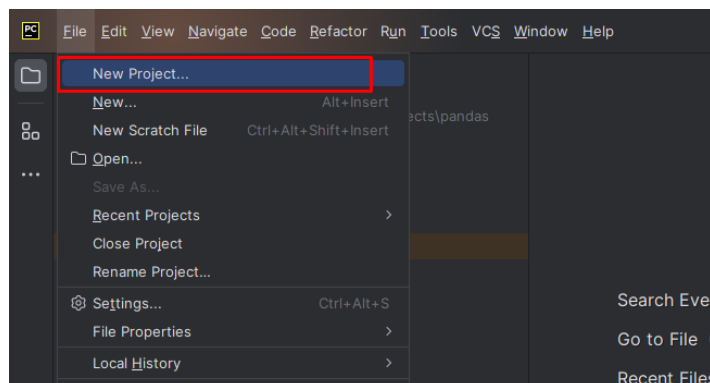


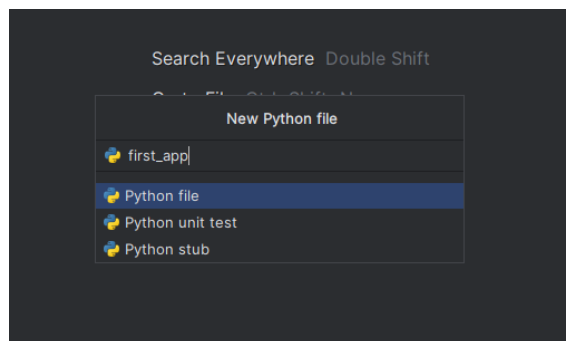
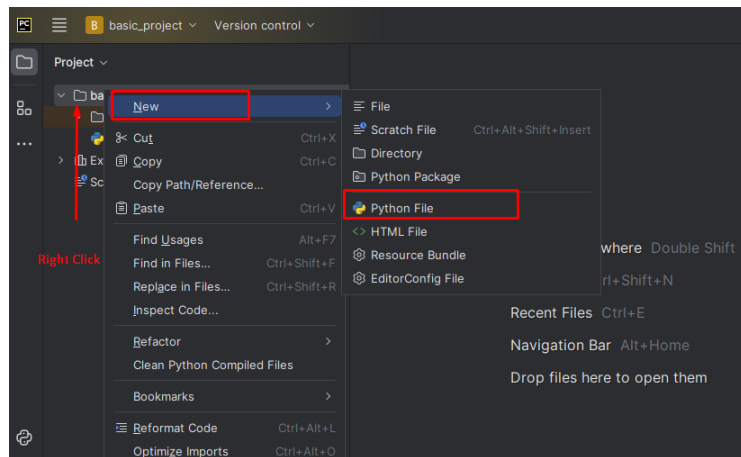
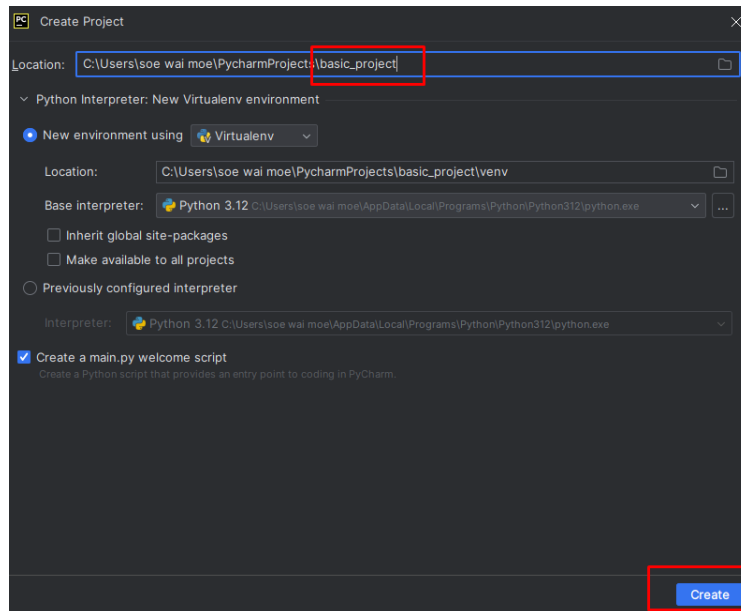


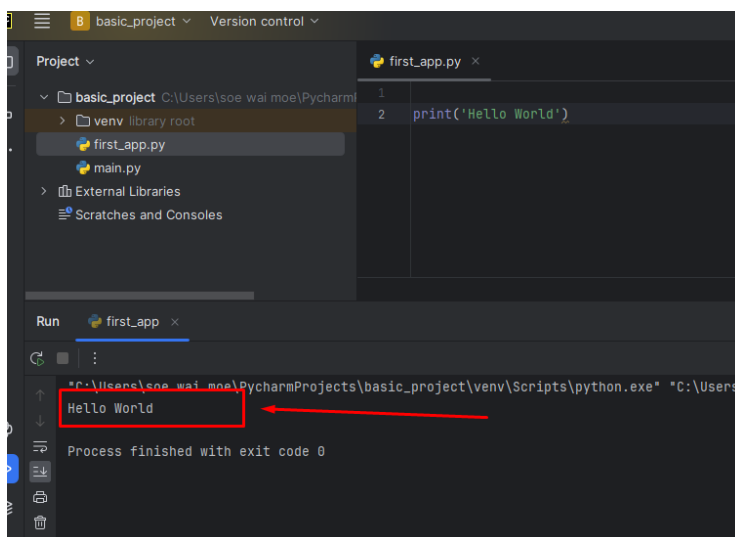
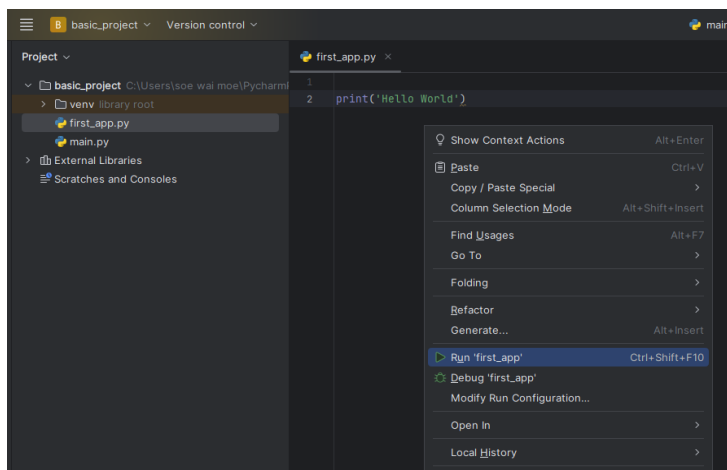
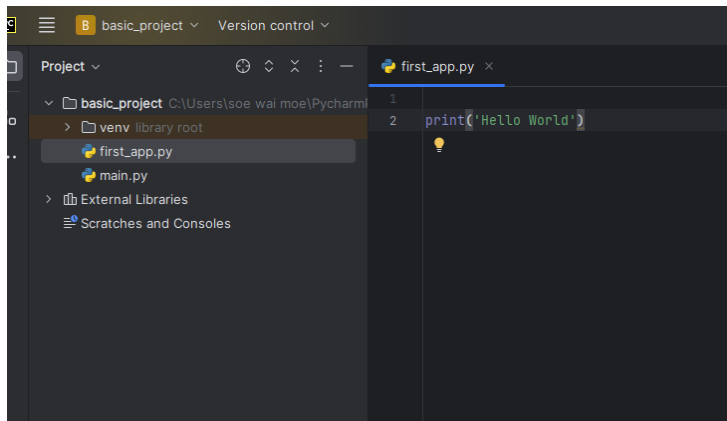


Python Installation is done. Pycharm installation is done. Pycharm is the Python IDE for data science and web development with intelligent code completion, on-the-fly error checking and quick-fixes.

Now you are ready to run your first python application. Open up your pycharm application, create a new project and run your first application as the followngs.



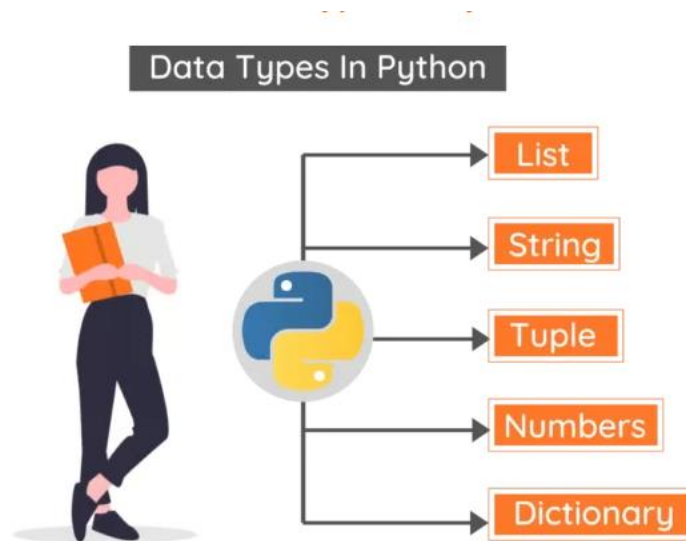




⇒ Congratulations! You have successfully run your first python application.

# Data Types

📺 video tutorial 2



## Python Data Types:

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple
- Mapping Type: dictionary
- Set Types: set
- Boolean Type: Boolean
- None Type: None

## Sample Data Types

<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset
<code>x = True</code>	bool
<code>x = b"Hello"</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview
<code>x = None</code>	NoneType

## Example 1

```
v1 = 5
v2=2.555
v3='hello'
v4= "John"
v5=True
v6=[1,2,3]
v7={'mgmg','su su'}
v8={'id':1,'name':'mgmg'}
v9=[1,"hello",33]
v10=None
```

```
print(type(v1))
print(type(v2))
print(type(v3))
print(type(v4))
print(type(v5))
print(type(v6))
print(type(v7))
print(type(v8))
print(type(v9))
print(type(v10))
```

Output:




```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'str'>
<class 'bool'>
<class 'list'>
<class 'set'>
<class 'dict'>
<class 'list'>
<class 'NoneType'>
```

---

## *Basic Calculations*

---

 video tutorial 3



## Example 1

#Before applying data types

```
print('Basic Calculations')
print('-----')
print('Add Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= num1 + num2

print('The answer is ',ans)
```

### Output:

```
Basic Calculations
-----
Add Numbers
-----
Enter Num1:12
Enter Num2:3
The answer is  123

Process finished with exit code 0
```

#After applying data type (numeric)

```
print('Basic Calculations')
print('-----')
print('Add Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= int(num1) + int(num2)

print('The answer is ',ans)
```

## Output

```
Basic Calculations
-----
Add Numbers
-----
Enter Num1:12
Enter Num2:3
The answer is  15

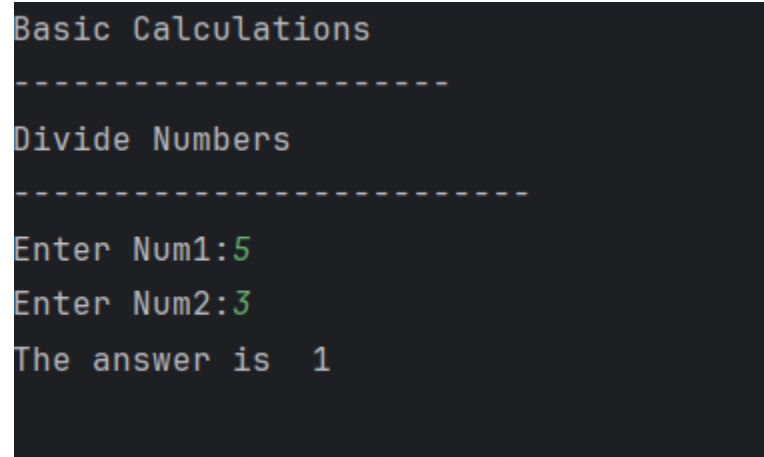
Process finished with exit code 0
```

## Practice 1

```
print('Basic Calculations')
print('-----')
print('Divide Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= int(num1) / int(num2)

print('The answer is ',int(ans))
```

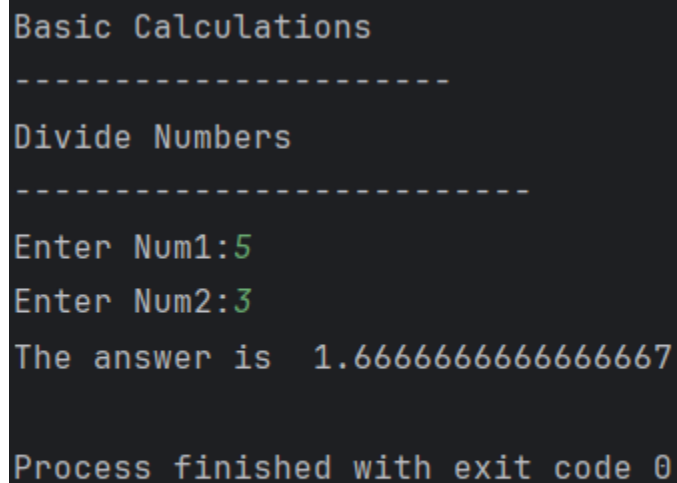
## Ouput



```
Basic Calculations
-----
Divide Numbers
-----
Enter Num1:5
Enter Num2:3
The answer is  1
```

```
print('Basic Calculations')
print('-----')
print('Divide Numbers  ')
print('-----')
num1=input('Enter Num1:')
num2=input('Enter Num2:')
ans= float(num1) / float(num2)

print('The answer is ',float(ans))
```



```
Basic Calculations
-----
Divide Numbers
-----
Enter Num1:5
Enter Num2:3
The answer is  1.6666666666666667

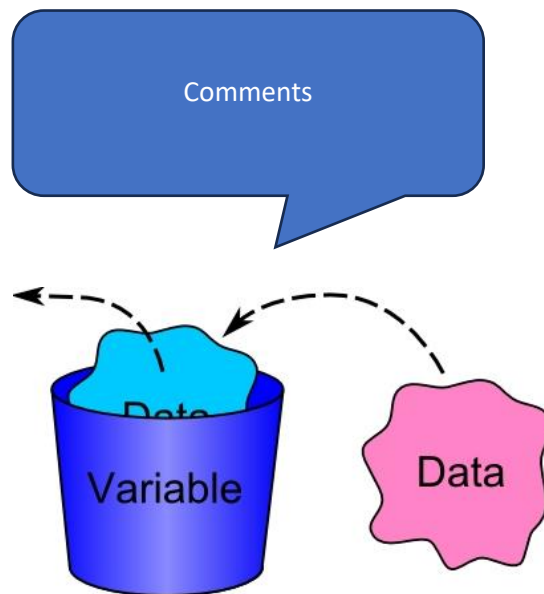
Process finished with exit code 0
```

---

## *Variables & Comments*

---

🎥 video tutorial 4



# Variables

Variables are containers for storing data values. Variables are names defined by a programmer. When it comes to variable naming convention, you may see the following three cases:

1. Pascal case  
- Examples: FirstName, LastName
2. Camel case  
- Examples: firstName, lastName
3. Snake case  
- Examples: first\_name, last\_name

The Pascal case is a naming convention similar to the Camel case. Where Pascal casing has the first letter of each word in uppercase and no spaces between words, Camel casing follows the same format with one exception – the first letter of the first word is lowercase, and subsequent words are capitalized. Snake case is designed to have underscores as a replacement for spaces. snake case is a convention that underscores separate words, such as “first\_input”.

## Example 1

first\_num => good practice

student\_name => good practice

1\_num => bad practice

@num => bad practice

## Comments

Comments can be used to explain Python code. Comments can be used to make the code more readable. Comments can be used to prevent execution when testing code.

1. Single line comment ( # )
2. Multiline comment ( """ multiple lines """ )



### Example

```
#This is a comment
```

```
#written in
```

```
#single line comment
```

```
print("Hello Northern City!")
```

```
"""
```

```
This is a comment
```

```
written in
```

```
multiline comment
```

```
"""
```

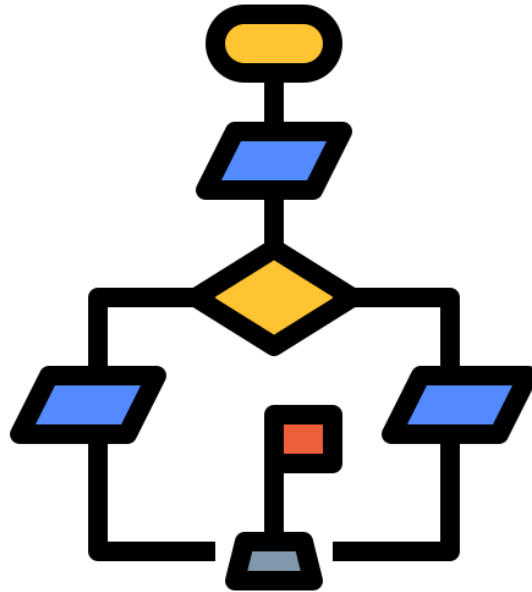
```
print("Hello World!")
```

---

## Conditional Statements

---

🎥 video tutorial 5



# Conditional Statements

- 1.If-else
- 2.If-elif
- 3.Nested if

## Syntax:

---

*If condition:*

*Truth Statement*

*else:*

*False Statement*

---

## Example 1:

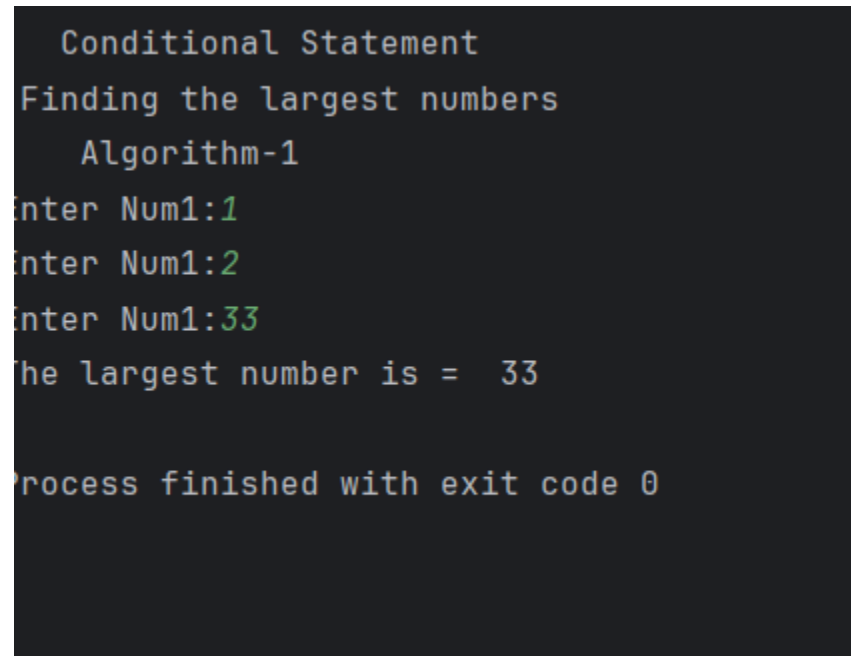
```
print(' Conditional Statement ')\nprint(' Finding the largest numbers ')\nprint(' Algorithm-1 ')\nnum1=int(input('Enter Num1:'))\nnum2=int(input('Enter Num2:'))\nnum3=int(input('Enter Num3:'))
```

```
if num1>num2:
    L=num1
else:
    L=num2

if num3>L:
    L=num3

print('The largest number is = ', L)
```

Output:



```
Conditional Statement
Finding the largest numbers
Algorithm-1
Enter Num1:1
Enter Num1:2
Enter Num1:33
The largest number is = 33
process finished with exit code 0
```

## Example 2

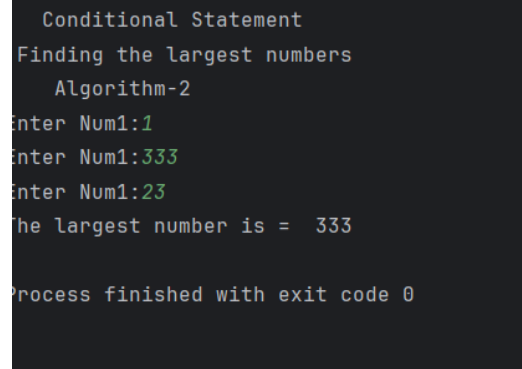
```
print(' Conditional Statement ')
print(' Finding the largest numbers ')
```

```
print(' Algorithm-2 ')
num1=int(input('Enter Num1:'))
num2=int(input('Enter Num2:'))
num3=int(input('Enter Num3:'))

if num1>num2 and num1>num3:
    L=num1
elif num2>num1 and num2>num3:
    L=num2
else:
    L=num3

print('The largest number is = ', L)
```

Output:



```
Conditional Statement
Finding the largest numbers
Algorithm-2
Enter Num1:1
Enter Num1:333
Enter Num1:23
The largest number is = 333

Process finished with exit code 0
```

## Example 3

```
print(' Conditional Statement ')
print(' Finding the largest numbers ')
print(' Algorithm-3 ')
num1=int(input('Enter Num1:'))
num2=int(input('Enter Num2:'))
num3=int(input('Enter Num3:'))
```

```
if num1>num2:
    if num1>num3:
        L=num1
    else:
        L=num3
else:
    if num2>num3:
        L=num2
    else:
        L=num3
```

```
print('The largest number is = ', L)
```

Output:

```
Conditional Statement
Finding the largest numbers
Algorithm-3
Enter Num1:333
Enter Num1:22
Enter Num1:55
The largest number is = 333

Process finished with exit code 0
```

## Practice

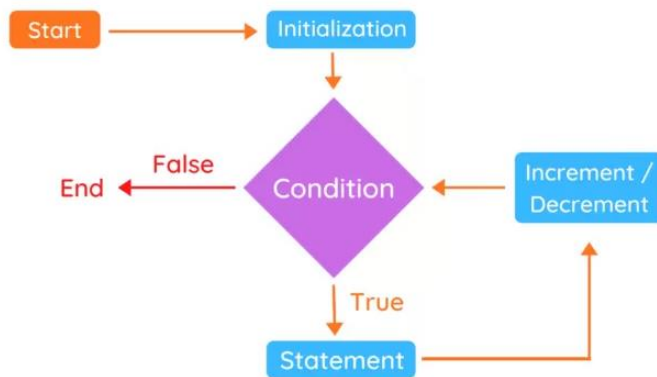
Practice the following exercise, you may use any of the algorithms above

```
Conditional Statement
Finding the largest numbers
Algorithm-3
Enter Num1:333
Enter Num2:2
Enter Num3:55
Enter Num4:5
The largest number is = 333

Process finished with exit code 0
```

## Loops

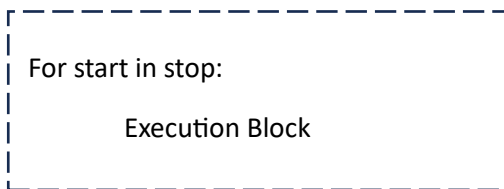
📺 video tutorial 6



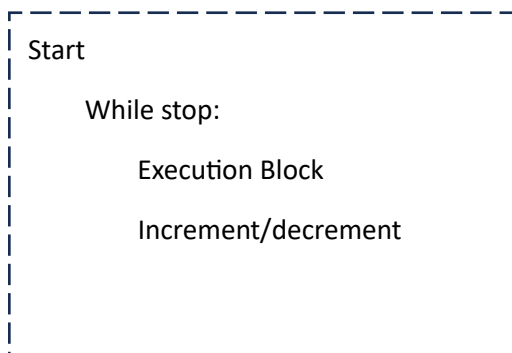


- For Loop
- While Loop

For Loop Syntax:



While Loop Syntax:



## Example 1

```
for i in range(5):  
    print('loop no=',i)
```

Output:

```
loop no= 0
loop no= 1
loop no= 2
loop no= 3
loop no= 4
```

## Example 2

```
j=0
while j<5:
    print('loop no=',j)
    j+=1  # j=j+1
```

Output:

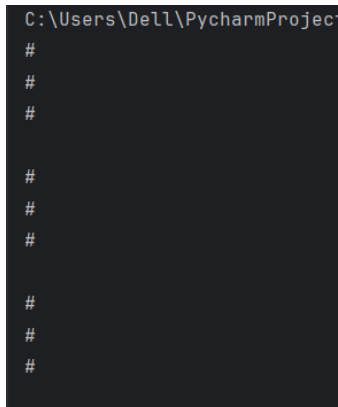
```
loop no= 0
loop no= 1
loop no= 2
loop no= 3
loop no= 4
```

## Example 3

```
for row in range(3):
    for col in range(3):
        print('#')
```

```
print() #new line (break line)
```

## Output



```
C:\Users\Dell\PycharmProject>
#
#
#
#
#
#
#
#
#
#
```

## Example 4

```
for row in range(10):
    for col in range(10):
        print('*',end='')
    print() #new line (break line)
```

## Output:

```
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****  
*****
```

## Example 5

```
for row in range(10):  
    for col in range(10):  
        if row==0 or row==9:  
            print('*',end='')  
  
        if row>0 and row<9:  
            if col==0 or col==9:  
                print('*',end='')  
            else:  
                print(' ',end='')  
  
    print()
```

Output:

```
*****
*       *
*       *
*       *
*       *
*       *
*       *
*       *
*       *
*       *
*****
```

## Example 6

```
for row in range(10):
    for col in range(10):
        if row==0 or row==4 or row==9:
            print('*',end='')
        if row>0 and row<4 or row>4 and row<9:
            if col==0 or col==9:
                print('*',end='')
            else:
                print(' ',end='')

    print()
```

Output:

```
*****
*       *
*       *
*       *
*****
*       *
*       *
*       *
*       *
*****
```

## Example 7

```
for row in range(11):
    for col in range(20):
        if row == 0 or row == 5 or row == 10:
            print('*', end='')

        if row>0 and row<5:
            if col==0 or col==4 or col==19:
                print('*', end='')
            else:
                print(' ', end='')

        if row > 5 and row < 10:
            if col==0 or col==19:
                print('*', end='')
            else:
                print(' ', end='')
    print()
```

Output:

```
*****
*   *   *
*   *   *
*   *   *
*   *   *
*****
*       *
*       *
*       *
*       *
*****
```

## Example 8

```
for row in range(11):
    for col in range(10):
        if row == 0 or row == 5 or row == 10:
            print('*', end='')

        if row>0 and row<5:
            if col==0 or col==9:
                print('*',end='')
            else:
                print(' ',end='')

        if row>5 and row<10:
            if col==0 or col==4 or col==9:
                print('*',end='')
            else:
                print(' ',end='')

    print()
```

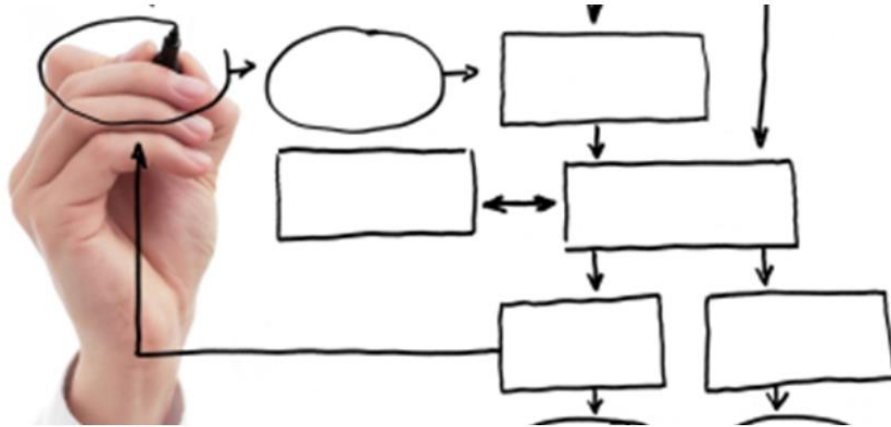
# Practice

```
*****
*      *          *
*      *          *
*      *          *
*****
*      *          *      *
*      *          *      *
*      *          *      *
*      *          *      *
*      *          *      *
*      *          *      *
*****
```



## Method

📺 video tutorial 7



## Method

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method. A method can return data as a result.

### Example 1

```
def exit():  
    print('Thanks for using my program ...')
```

```
def display(ans):  
    print('The answer is = ', ans)
```

```
def add_num(num1, num2):  
    return num1 + num2
```

```
def sub_num(num1, num2):  
    return num1 - num2
```

```
def multiply_num(num1, num2):  
    return num1 * num2
```

```
def divide_num(num1, num2):  
    return num1 / num2
```

```
def get_input():  
    return int(input("Enter a number:"))
```

```
def add():  
    x = get_input()  
    y = get_input()
```

```
ans = add_num(x, y)
display(ans)
```

```
def sub():
    x = get_input()
    y = get_input()

    ans = sub_num(x, y)
    display(ans)
```

```
def multiply():
    x = get_input()
    y = get_input()

    ans = multiply_num(x, y)
    display(ans)
```

```
def divide():
    x = get_input()
    y = get_input()

    ans = divide_num(x, y)
    display(ans)
```

```
def get_choice():
    return int(input('Enter Choice:'))
```

```
def menu():
    print(' Method Example 1 ')
    print(' ----- ')
    print(' [1] Add Numbers ')
    print(' [2] Subtract Numbers ')
    print(' [3] Multiple Numbers ')
    print(' [4] Divide Numbers ')
    print(' ----- ')
    print(' [5] Exit Program ')
    print('-----')
```

```
def main():
    loop = True
    while loop:
        menu()
        ch = get_choice()

        if ch == 1:
            add()
        elif ch == 2:
            sub()
        elif ch == 3:
            multiply()
        elif ch == 4:
            divide()
        elif ch == 5:
            exit()
            loop = False
        else:
            print('invalid choice...')

if __name__ == '__main__':
    main()
```

## Output:

```
Method Example 1
-----
[1] Add Numbers
[2] Subtract Numbers
[3] Multiple Numbers
[4] Divide Numbers
-----
[5] Exit Program
-----
Enter Choice:
```

## Example 2

```
pi=3.14159
def exit():
    print('Thanks for using my program ...')
def back():
    print('Back to Main Menu...')

def display(ans):
    print('The answer is = ', ans)

def get_circle_area(r):
    return pi * r * r
def get_circle_circumference(r):
    return 2 * pi * r

def get_triangle_area(base, height):
    return base * height * 0.5

def get_triangle_volume(base, height):
    return (base * height) / 3

def circle_circumference():
    r = get_radius()

    ans = get_circle_circumference(r)
    display(ans)

def circle_area():
    r = get_radius()

    ans = get_circle_area(r)
    display(ans)
```

```
def circle():
    loop=1
    while loop:
        circle_menu()
        ch=get_choice()
        if ch==1:
            circle_area()
        elif ch==2:
            circle_circumference()
        elif ch==3:
            back()
            loop=0
        else:
            print('invlid choice...')
```

```
def triangle():
    loop=1
    while loop:
        triangle_menu()
        ch=get_choice()
        if ch==1:
            triangle_area()
        elif ch==2:
            triangle_volume()
        elif ch==3:
            back()
            loop=0
        else:
            print('invlid choice...')
```

```
def get_radius():
    return float(input('Enter Radius:'))
def get_base():
    return float(input('Enter Base:'))
def get_height():
    return float(input('Enter Height:'))
def triangle_area():
    b = get_base()
    h = get_height()

    ans = get_triangle_area(b, h)
```

```
display(ans)

def triangle_volume():
    b = get_base()
    h = get_height()

    ans = get_triangle_volume(b, h)
    display(ans)

def get_choice():
    return int(input('Enter Choice:'))

def triangle_menu():
    print(' Triangle MENU ')
    print(' ----- ')
    print(' [1] Triangle Area ')
    print(' [2] Triangle Volume ')
    print('-----')
    print(' [3] Back  ')
    print('-----')

def circle_menu():
    print(' Circle MENU ')
    print(' ----- ')
    print(' [1] Circle Area ')
    print(' [2] Circle Circumference ')
    print('-----')
    print(' [3] Back  ')
    print('-----')

def main_menu():
    print(' Method Example-2 ')
    print(' ----- ')
    print(' [1] Circle ')
    print(' [2] Triangle ')
    print('-----')
    print(' [3] Exit Program ')
    print('-----')

def main():
    loop = 1
    while loop:
```

```
main_menu()
ch = get_choice()

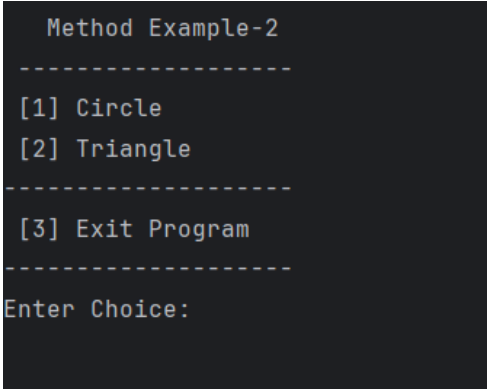
if ch == 1:
    circle()

elif ch == 2:
    triangle()

elif ch == 3:
    exit()
    loop = 0
else:
    print('invalid choice...')

if __name__ == '__main__':
    main()
```

Output:



```
Method Example-2
-----
[1] Circle
[2] Triangle
-----
[3] Exit Program
-----
Enter Choice:
```




## Practice

```
Method Practice
-----
[1] Show Largest Number
[2] Show Smallest Number
-----
[3] Exit Program
-----
Enter Choice:
```

---

## *Class (OOP)*

---

 video tutorial 8

# CLASS {OOP}

# Class

Python is an object oriented programming language. Almost everything in Python is an object, with its properties and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

- objects
- Inheritance
- Polymorphism
- encapsulation

## Objects

An object consists of:

- **State:** It is represented by the attributes of an object. It also reflects the properties of an object.
- **Behavior:** It is represented by the methods of an object. It also reflects the response of an object to other objects.
- **Identity:** It gives a unique name to an object and enables one object to interact with other objects.

## Example 1

```
class Dog:
```

```
    # class attribute
    attr1 = "mammal"
```

```
    # Instance attribute
    def __init__(self, name):
        self.name = name
```

```
# Object instantiation
Dog1 = Dog("Aung Net")
Dog2 = Dog("Mae Lone")
```

```
# Accessing class attributes
print("Aung Net is a {}".format(Dog1.__class__.attr1))
```

```
print("Mae Lone is also a {}".format(Dog2.__class__.attr1))
```

```
# Accessing instance attributes
```

```
print("My name is {}".format(Dog1.name))
```

```
print("My name is {}".format(Dog2.name))
```

## Output:

Aung Net is a mammal

Mae Lone is also a mammal

My name is Aung Net

My name is Mae Lone

## Example 2

```
class Dog:
```

```
    # class attribute
```

```
    attr1 = "mammal"
```

```
    # Instance attribute
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
    def speak(self):
```

```
        print("My name is {}".format(self.name))
```

```
# Driver code
```

```
# Object instantiation
```

```
Dog1 = Dog("Aung Net")
```

```
Dog2 = Dog("Phoe Kyar")
```

```
# Accessing class methods
```

```
Dog1.speak()
```

```
Dog2.speak()
```

Output:

My name is Aung Net

My name is Phoe Kyar

# Inheritance

In Python object oriented Programming, Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class. The benefits of inheritance are:

- It represents real-world relationships well.
- It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

In this example, we have two classes i.e. Person (parent class) and Employee (Child Class). The Employee class inherits from the Person class. We can use the methods of the person class through the employee class as seen in the display function in the above code. A child class can also modify the behavior of the parent class as seen through the details() method.

## Example 1

# Python code to demonstrate how parent constructors  
# are called.

# parent class  
class Person(object):

    # \_\_init\_\_ is known as the constructor  
    def \_\_init\_\_(self, name, idnumber):  
        self.name = name  
        self.idnumber = idnumber

    def display(self):  
        print(self.name)  
        print(self.idnumber)

    def details(self):  
        print("My name is {}".format(self.name))

```
print("IdNumber: {}".format(self.idnumber))

# child class
class Employee(Person):
    def __init__(self, name, idnumber, salary, position):
        self.salary = salary
        self.position = position

    # invoking the __init__ of the parent class
    Person.__init__(self, name, idnumber)

    def details(self):
        print("My name is {}".format(self.name))
        print("Salary: {}".format(self.salary))
        print("Position: {}".format(self.position))

# creation of an object variable or an instance
a = Employee('Mg Mg', 886012, 850000, "Senior Developer")

# calling a function of the class Person using
# its instance
a.display()
a.details()
```

Output:

```
Mg Mg
886012
My name is Mg Mg
Salary: 850000
Position: Senior Developer
```

## Polymorphism

In object oriented Programming Python, Polymorphism simply means having many forms. For example, we need to determine if the given species of birds fly or not, using polymorphism we can do this using a single function.

## Example 1

```
class Bird:

    def intro(self):
        print("There are many types of birds.")

    def flight(self):
        print("Most of the birds can fly but some cannot.")
```

```
-----

class sparrow(Bird):

    def flight(self):
        print("Sparrows can fly.")
```

```
-----

class ostrich(Bird):

    def flight(self):
        print("Ostriches cannot fly.")
```

```
-----

obj_bird = Bird()
obj_spr = sparrow()
obj_ost = ostrich()

obj_bird.intro()
obj_bird.flight()

obj_spr.intro()
```

```
obj_spr.flight()
```

```
obj_ost.intro()
```

```
obj_ost.flight()
```

## Output:

There are many types of birds.

Most of the birds can fly but some cannot.

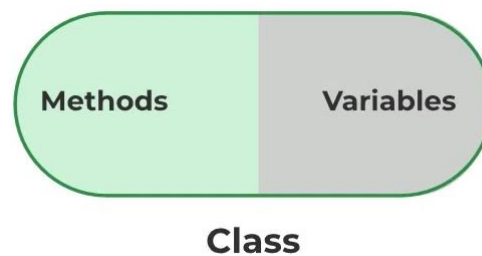
There are many types of birds.

Sparrows can fly.

There are many types of birds.

Ostriches cannot fly.

## Encapsulation



In Python object oriented programming, Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as private variables.

Python provides three types of access modifiers private, public, and protected.

- **Public Member:** Accessible anywhere from outside class.
- **Private Member:** Accessible within the class
- **Protected Member:** Accessible within the class and its sub-classes



```
class Employee:
    def __init__(self, name, project):
        self.name = name
        self.project = project
    def work(self):
        print(self.name, 'is working on', self.project)
```

} Data Members

Method {

Wrapping data and the methods that work on data  
within one unit

**Class** (Encapsulation)

## Public Member

Public data members are accessible within and outside of a class. All member variables of the class are by default public.

## Example

```
class Employee:
    # constructor
    def __init__(self, name, salary, project):
        # data members
        self.name = name
        self.salary = salary
        self.project = project

    # method
    # to display employee's details
    def show(self):
        # accessing public data member
        print("Name: ", self.name, 'Salary:', self.salary)

    # method
```

```
def work(self):
    print(self.name, 'is working on', self.project)

# creating object of a class
emp = Employee('Mg Mg', 8000, 'NC Web')

# calling public method of the class
emp.show()
emp.work()
```

Output:

```
Name: Jessa Salary: 8000
Mg Mg is working on NC Web
```

## Private Member

We can protect variables in the class by marking them private. To define a private variable add two underscores as a prefix at the start of a variable name.

Private members are accessible only within the class, and we can't access them directly from the class objects.

## Example

```
class Employee:
    # constructor
    def __init__(self, name, salary):
        # public data member
        self.name = name
        # private member
        self.__salary = salary

# creating object of a class
emp = Employee('Mg Mg', 1000000)

# accessing private data members
print('Salary:', emp.__salary)
```

## Output:

AttributeError: 'Employee' object has no attribute '\_\_salary'

## Rewrite the Example

```
class Employee:
    # constructor
    def __init__(self, name, salary):
        # public data member
        self.name = name
        # private member
        self.__salary = salary

    # public instance methods
    def show(self):
        # private members are accessible from a class
        print("Name: ", self.name, 'Salary:', self.__salary)

# creating object of a class
emp = Employee('Mg Mg', 1000000)

# calling public method of the class
emp.show()
```

Output:

Name: Mg Mg Salary: 1000000

## Protected Member

Protected members are accessible within the class and also available to its sub-classes. To define a protected member, prefix the member name with a single underscore \_.

Protected data members are used when you implement inheritance and want to allow data members access to only child classes.

## Example

```
# base class
```

```
class Company:
    def __init__(self):
        # Protected member
        self._project = "NC Web"

# child class
class Employee(Company):
    def __init__(self, name):
        self.name = name
        Company.__init__(self)

    def show(self):
        print("Employee name :", self.name)
        # Accessing protected member in child class
        print("Working on project :", self._project)

c = Employee("Mg Mg")
c.show()

# Direct access protected data member
print('Project:', c._project)
```

## Output:

```
Employee name : Mg Mg
Working on project : NC Web
Project: NC Web
```

## Abstraction

Abstraction can be achieved in Python using abstract base classes (ABCs) provided by the "abc" module. These classes allow you to define abstract methods that subclasses must implement.

Shape (Parent Class)

Unimplemented methods: area, perimeter

```
def area():  
    Pass  
def perimeter():  
    pass
```

```
Def area():  
    Return 3.14*r*r  
Def color():  
    Return 2 * 3.14 * r
```

Circle  
( Child Class)

```
Def area():  
    Return l*w  
Def perimeter():  
    Return 2(l*w)
```

Rectangle  
( Child Class)

## Example

```
from abc import ABC, abstractmethod
```

```
class Shape(ABC):  
    @abstractmethod  
    def area(self):  
        pass
```

```
    @abstractmethod  
    def perimeter(self):  
        pass
```

```
class Circle(Shape):  
    def __init__(self, radius):  
        self.radius = radius
```

```
    def area(self):  
        return 3.14 * self.radius * self.radius
```

```
def perimeter(self):
    return 2 * 3.14 * self.radius

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * ( self.length * self.width )

circle = Circle(3)
print('Circle Area =',circle.area())
print('Circle Perimeter =', circle.perimeter())

rect= Rectangle(3,2)
print('Rectangle Area = ',rect.area())
print('Rectcangle Perimeter = ',rect.perimeter())
```

## Output:

```
Circle Area = 28.259999999999998
Circle Perimeter = 18.84
Rectangle Area = 6
Rectcangle Perimeter = 12
```

# Practice

-----  
Class Practice

Sein Gay Har

-----  
MENU

-----  
[1] Add New Employee

[2] Show Employees

-----  
[3] Exit Program  
-----

class:

employee

id,name,position,salary,phone,address


list:

employee\_list=[]

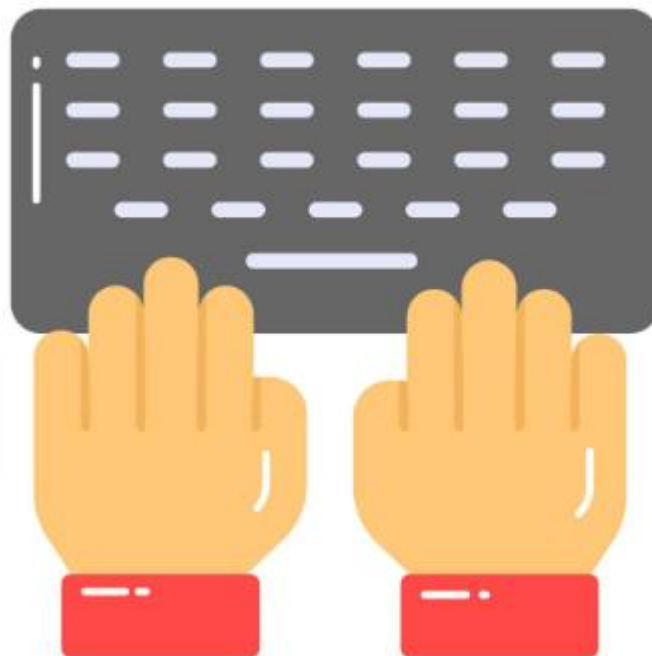
---

## *Lists*

---

 video tutorial 9

[10,20,30,40,50]





# Lists

Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage. Lists are created using square brackets:

- Access List items
- List Compensations
- Lists functions

## ⇒ Access List item

### Example 1

```
thislist = ["apple", "banana", "cherry"]  
print(thislist[1])
```

Output:

Banana

### Example 2

#### Negative Indexing

=> Negative indexing means start from the end

=> -1 refers to the last item, -2 refers to the second last item etc.

```
thislist = ["apple", "banana", "cherry"]  
print(thislist[-1])
```

Output:

cherry

## Example 3

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5])  
print(thislist[2:])  
print(thislist[1:-1])  
print(thislist[::-1])
```

Output:

```
['cherry', 'orange', 'kiwi']  
['cherry', 'orange', 'kiwi', 'melon', 'mango']  
['banana', 'cherry', 'orange', 'kiwi', 'melon']  
['mango', 'melon', 'kiwi', 'orange', 'cherry', 'banana', 'apple']  
['apple', 'cherry', 'kiwi']
```

## ⇒ Lists Compensation

### Syntax:

```
newlist = [expression for item in iterable if condition == True]
```

## Example 1

⇒ Ordinary Lists:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []
```

```
for x in fruits:
```

```
if "a" in x:
    newlist.append(x)

print(newlist)
```

⇒ Rewrite to Compensation List:

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "a" in x]
print(newlist)
```

Output:

```
['apple', 'banana', 'mango']
```

## Example 2

```
even_list=newlist = [x for x in range(20) if x%2==0]
print(even_list)
```

Output:

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
odd_list=newlist = [x for x in range(20) if x%2==1]
print(odd_list)
```

Output:

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

## Example 3

#2D Array

```
arr = [['0' for i in range(10)] for j in range(10)]
```

```
for row in arr:
```

```
    print(row)
```

Output:

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

```
['0', '0', '0', '0', '0', '0', '0', '0', '0', '0']
```

## Example 4

```
students=[  
    ("mg mg",22,"male"),  
    ("su su",18,"female"),  
    ("tun tun",14,"male"),  
    ("kyaw kyaw",12,"male"),  
    ("soe soe",15,"female"),  
    ("sai sai",21,"male"),  
    ("moe moe",25,"female")  
]
```

```
female_list=[rec for rec in students if rec[2]=='female']
```

```
for student in female_list:  
    print(student)
```

## Output:

```
('su su', 18, 'female')  
('soe soe', 15, 'female')  
('moe moe', 25, 'female')
```

## ⇒ Lists Functions

### Example 1

```
number_list=[2,1,23,45,90]  
  
print("list positional values...")  
print(" index 0 =",number_list[0])  
print(" index 1 =",number_list[1])  
print(" index 2 =",number_list[2])  
  
print("=====")  
print("generate list values using loop index")  
  
for i in range(len(number_list)):  
    print(number_list[i])  
  
print("=====")  
print("generate list values using list content")  
  
for num in number_list:  
    print(num)
```

## Output:

```
list positional values...
index 0 = 2
index 1 = 1
index 2 = 23
=====
generate list values using loop index
2
1
23
45
90
=====
generate list values using list content
2
1
23
45
90
```

## Example 2

```
student_list=["mg mg","aung aung","su su","tun tun"]

#display all studnets
print("Display all student list")
for student in student_list:
    print(student)

#add new student

new_student="bo bo"
student_list.append(new_student)

#display all studnets
print("Display all student list after adding new student bo bo")
for student in student_list:
    print(student)

#edit student name
```

```
#remove current data
student_list.remove(student_list[4])
#update new data
update_student="bo bo aung"
student_list.insert(4,update_student)

# display all studnets
print("Display all student list after updating student bo bo to bo bo aung")
for student in student_list:
    print(student)

# delete data
print("delete last student")
student_list.pop()

# display all studnets
print("Display all student list after deleting last index")
for student in student_list:
    print(student)

# delete data
print("delete student using del command")
del student_list[1]

# display all studnets
print("Display all student list after deleting an item using del keyword")
for student in student_list:
    print(student)
```

## Output:

```
Display all student list
mg mg
aung aung
su su
tun tun
Display all student list after adding new student bo bo
mg mg
aung aung
```

```
su su
tun tun
bo bo
Display all student list after updating student bo bo to bo bo aung
mg mg
aung aung
su su
tun tun
bo bo aung
delete last student
Display all student list after deleting last index
mg mg
aung aung
su su
tun tun
delete student using del command
Display all student list after deleting an item using del keyword
mg mg
su su
tun tun
```

## Example 3

```
#initialization of lists
contacts=[]

def exit_program():
    print('Exit Program Now...Bye Bye...')

def view_number():
    for number in contacts:
        temp=str(number).split('-')
        print('-----')
        print('Name : ',temp[0])
        print('Address :',temp[1])
        print('-----')
```



```
def get_name():
    return input('Enter New Name:')

def get_address():
    return input('Enter New Address:')

def add_new_contact():
    new_name=get_name()
    new_address=get_address()

    new_contact=new_name+"-"+new_address

    contacts.append(new_contact)

    print('Saving Success')

def get_choice():
    return int(input('Enter Choice:'))
def main_menu():
    print(' View Address  ')
    print('-----')
    print('[1] Add New Contact  ')
    print('[2] View Book  ')
    print('-----')
    print('[3] Exit  ')
    print('-----')

def main():
    loop=1
    while loop:
        main_menu()

        ch=get_choice()
        if ch==1:
            add_new_contact()
        elif ch==2:
            view_book()
        elif ch==3:
            exit_program()
        loop=0
```

```
else:  
    print('invalid choice...')
```

```
if __name__ == '__main__':  
    main()
```

## Output:

```
-----  
    Address Book  
-----  
[1] Add New Contact  
[2] View Book  
-----  
[3] Exit  
-----  
Enter Choice:1  
Enter New Name:mg mg  
Enter New Address:heldan  
Saving Success
```

```
-----  
    Address Book  
-----  
[1] Add New Contact  
[2] View Book  
-----  
[3] Exit  
-----  
Enter Choice:2  
-----  
Name : mg mg  
Address : heldan  
-----
```

## Practice

```
Phone Book
-----
[1] Add New Contact
[2] View Contacts
-----
[3] Exit
-----
Enter Choice:
```

---

## *Tuples*

---

📺 video tutorial 10

(1, 'python', 100)



# Tuples

Tuples are used to store multiple items in a single variable. Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage. A tuple is a collection which is ordered and unchangeable. Tuples are written with round brackets.

## Example 1

```
student1=(1,'mg mg',22,'heldan')
student2=(2,'su su',22,'heldan')
print(student1)
```

```
print("id =",student1[0])
print("name =",student1[1])
print("age =",student1[2])
```

```
students=[]
```

```
students.append(student1)
students.append(student2)
```

```
print(students)
```

```
my_student_array=[]
```

```
my_student_array[1]=student1
my_student_array[2]=student2
```

```
print(my_student_array)
```

```
print(my_student_array[1])
```

```
print(my_student_array[2])
```

## Output:

```
(1, 'mg mg', 22, 'heldan')
id = 1
name = mg mg
age = 22
[(1, 'mg mg', 22, 'heldan'), (2, 'su su', 22, 'heldan')]
{1: (1, 'mg mg', 22, 'heldan'), 2: (2, 'su su', 22, 'heldan')}
(1, 'mg mg', 22, 'heldan')
(2, 'su su', 22, 'heldan')
```

## Example 2

```
class Account():
    def __init__(self,id,name,nrc,amount):
        self.id=id
        self.name=name
        self.nrc=nrc
        self.amount=amount

    def set_id(self,id):
        self.id=id
    def get_id(self):
        return self.id

    def set_name(self,name):
        self.name=name

    def get_name(self):
        return self.name

    def set_nrc(self,nrc):
        self.nrc=nrc
    def get_nrc(self):
        return self.nrc
    def set_amount(self,amount):
        self.amount=amount

    def get_amount(self):
```

```
        return self.amount

    def display(self):
        print('-----')
        print('Id =',self.id)
        print('Name = ',self.name)
        print('Nrc =',self.nrc)
        print('Amount = ',self.amount)
        print('-----')
```

YomaBankApp.py

```
accounts=[]

def get_withdraw_amt():
    return int(input('Enter Withdraw amount:'))
def get_deposit():
    return int(input('Enter deposit amount:'))

def close_account():
    print('Close Account...')

    found=0
    found_index=-1
    i=0

    deposit_id=get_id()
    for account in accounts:
        if deposit_id==account[0]:
            found=1
            found_index=i
            break
        i+=1

    if found==1:
        del accounts[found_index]
        print('account successfully closed...')
    else:
        print('Sorry..account id not found..try again...')

def withdraw_fund():
```

```
print('Deposit Fund...')

found=0
found_index=-1
i=0

deposit_id=get_id()
for account in accounts:
    if deposit_id==account[0]:
        found=1
        found_index=i
        break
    i+=1

if found==1:
    print('Account Found...')
    #since the object found is tuple we cannot change the value
    #that is why we need to convert it into list as the following
    curr_account=list(accounts[found_index])
    print('Current Amount is = ',curr_account[3])
    curr_amount=curr_account[3]
    withdraw_amt=get_withdraw_amt()
    if withdraw_amt> curr_amount:
        print('Sorry...Insufficient fund...try again...')
        print('Your current amount is ony...',curr_amount)
    else:
        update_amount=curr_amount - deposit_amt
        update_account=(curr_account[0],curr_account[1],curr_account[2],update_amount)
        del accounts[found_index]
        accounts.insert(found_index,update_account)

    print('Deposit Success...')

else:
    print('Sorry..Account Id not found...try agian...')

def deposit():
    print('Deposit Fund...')

    found=0
    found_index=-1
    i=0

    deposit_id=get_id()
```



```
for account in accounts:
    if deposit_id==account[0]:
        found=1
        found_index=i
        break
    i+=1

if found==1:
    print('Account Found...')
    #since the object found is tuple we cannot change the value
    #that is why we need to convert it into list as the following
    curr_account=list(accounts[found_index])
    print('Current Amount is = ',curr_account[3])
    curr_amount=curr_account[3]
    deposit_amt=get_deposit()
    update_amount=curr_amount + deposit_amt
    update_account=(curr_account[0],curr_account[1],curr_account[2],update_amount)
    del accounts[found_index]
    accounts.insert(found_index,update_account)

    print('Deposit Success...')

else:
    print('Sorry..Account Id not found...try agian...')

def display(acc):
    print('-----')
    print('Account Id = ',acc[0])
    print('Account Name = ', acc[1])
    print('NRC = ',acc[2])
    print('Amount = ', acc[3])
    print('-----')
def exit_program():
    print('Exit Program Now..Bye Bye...')
def view_accounts():
    for account in accounts:
        display(account)
def get_id():
    return input('Enter Id:')

def get_name():
    return input('Enter Name:')
```

```
def get_nrc():
    return input('Enter Nrc:')

def get_amount():
    return int(input('Enter Amount:'))

def create_account():
    id=get_id()
    name=get_name()
    nrc=get_nrc()
    amount=get_amount()

    account=(id,name,nrc,amount)

    accounts.append(account)

    print('Saving Success...')

def get_choice():
    return int(input('Enter Choice:'))

def main_menu():
    print(' Yoma Bank ')
    print('-----')
    print('  MENU  ')
    print('-----')
    print('[1] Create Account ')
    print('[2] View Accounts ')
    print('[3] Deposit Fund ')
    print('[4] Deposit Fund ')
    print('[5] Deposit Fund ')
    print('-----')
    print('[6] Exit Program ')
    print('-----')

def main():
    loop=1
    while loop:
        main_menu()
        ch=get_choice()

        if ch==1:
```

```
        create_account()
    elif ch==2:
        view_accounts()
    elif ch==3:
        deposit_fund()
    elif ch==4:
        withdraw_fund()
    elif ch==5:
        close_account()
    elif ch==6:
        exit_program()
    else:
        print('invalid choice...')
```

```
if __name__=='__main__':
    main()
```

## Output:

```
-----
Tuples Exercise
Yoma Bank
-----
      MENU
-----
[1] Create Account
[2] View Accounts
[3] Deposit Fund
[4] Deposit Fund
[5] Deposit Fund
-----
[6] Exit Program
-----
Enter Choice:
```

# Practice

-----  
SBT Car Sales  
-----

[1] Add New Car Record

[2] Show Car List

[3] Sell Car

[4] Delete Car Record  
-----

[5] Exit  
-----

Car.py

Car Class:


Id,name,brand,price,model,sale\_status

SBTApp.py

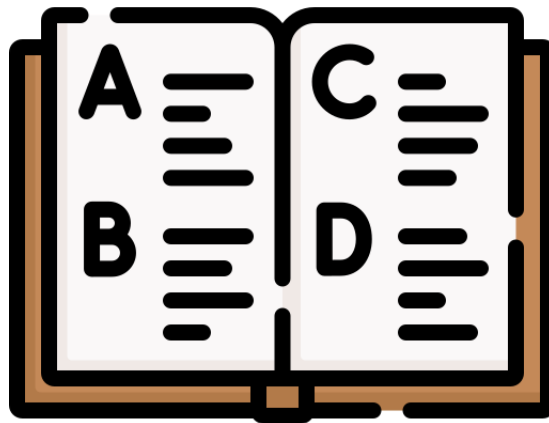
---

## Dictionary

---

 video tutorial 11

`{'A': 'First Letter', 'Language': 'English'}`



# Dictionary

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is ordered\*, changeable and do not allow duplicates. Dictionaries are written with curly brackets, and have keys and values.

## Syntax:

```
dict_var = {key1 : value1, key2 : value2, .....}
```

## Example 1

```
product={'id':1,'name':'coke','price':1200}
```

```
print('product id =',product['id'])
print('product name =',product['name'])
print('product price =',product['price'])
```

```
product_list=[1:{'id':1,'name':'coke','price':1200},2:{'id':2,'name':'apple','price':2000},3:{'id':3,'name':'pepsi','price':3000}]
```

```
print('retrieve object data by keys ')
print('product no 1 = ',product_list[1])
print('product no 2 = ',product_list[2])
```

```
print('retrieve object details ')
print('product no 1 = ',product_list[1]['name'])
print('product no 2 = ',product_list[3]['name'])
```

Output:

```
product id = 1
product name = coke
product price = 1200
retrieve object data by keys
product no 1 = {'id': 1, 'name': 'coke', 'price': 1200}
product no 2 = {'id': 2, 'name': 'apple', 'price': 2000}
retrieve object details
product no 1 = coke
product no 2 = pepsi
```

## Example 2

Item.py

```
class item():
    #constructor
    def __init__(self,id,name,price):
        self.id=id
        self.name=name
        self.price=price
    #getters and setters
    def get_id(self):
        return self.id

    def set_id(self,id):
        self.id=id

    def get_name(self):
        return self.name

    def set_name(self,name):
        self.name=name

    def get_price(self):
        return self.price

    def set_price(self,price):
        self.price=price

    def display(self):
        print('-----')
        print('Id : ',self.id)
        print('Name : ',self.name)
        print('Price : ',self.price)
        print('-----')
```

SeinGayHarApp.py

```
from Item import item
```

```
#dictionary declaration
item_list={}

def exit_program():
    print('Exit Program Now..Bye Bye...')
def view_items():
    for item in item_list.values():
        item.display()
def get_id():
    return input('Enter Id:')

def get_name():
    return input('Enter Name:')

def get_price():
    return input('Enter Price:')

def add_new_item():
    id=get_id()
    name=get_name()
    price=get_price()

    new_item=item(id,name,price)

    item_list[id]=new_item

    print('Saving Success...')

def get_choice():
    return int(input('Enter Choice:'))
def main_menu():
    print('----- ')
    print(' Dictionary Example 2')
    print(' Sein Gay Har ')
    print('-----')
    print(' MENU ')
    print('-----')
    print('[1] Add New Item ')
    print('[2] Show Items ist ')
    print('-----')
    print('[3] Exit Program ')
    print('-----')
```



```
def main():
    loop=1
    while loop:
        main_menu()

        ch=get_choice()

        if ch==1:
            add_new_item()
        elif ch==2:
            view_items()
        elif ch==3:
            exit_program()
            loop=0
        else:
            print('invalid choice...')

if __name__=='__main__':
    main()
```

Output:

```
-----
Dictionary Example 2
  Sein Gay Har
-----
      MENU
-----
[1] Add New Item
[2] Show Items ist
-----
[3] Exit Program
-----
Enter Choice:
```

## Practice

```
-----  
Dictionary Practice  
  Starbuck Coffee  
-----  
      MENU  
-----  
[1] Add New Item  
[2] View Items  
[3] Update Price by ID  
-----  
[4] Exit Program  
-----  
Enter Choice:
```

Coffee.py


- Id,category,name,price

StarbuckApp.py

---

## *File*

---

 video tutorial 12



# File

The key function for working with files in Python is the `open()` function. The `open()` function takes two parameters; filename, and mode. There are four different methods (modes) for opening a file:

## Example 1

```
#File Writing or Saving File
student_list=["mg mg","su su","tun tun","aung aung"]

file=open('students.txt','w')

for student in student_list:
    file.write(str(student)+'\n')
```

## Example 2

```
#File Reading
file_data=open('students.txt','r')

lines=file_data.readlines()

for line in lines:
    print(line.rstrip('\n'))
```

## Example 3

```
def show_items():
    data=open('items.txt','r')
```

```
lines=data.readlines()
count=0
for line in lines:
    if count%3==0:
        print('-----')
        count+=1

    print(line.rstrip('\n'))

print('-----')
def get_price():
    return input('Enter Price:')
def get_name():
    return input('Enter Name:')
def get_id():
    return input('Enter Id:')
def save_data(id,name,price):
    file=open('items.txt','a')
    file.write(str(id)+'\n')
    file.write(str(name)+'\n')
    file.write(str(price)+'\n')
def add_new_item():
    id=get_id()
    name=get_name()
    price=get_price()

    save_data(id,name,price)

    print('Saving Success....')

def get_choice():
    return int(input('Enter Choice:'))

def main_menu():
    print('  File Example 3  ')
    print(' ----- ')
    print('  City Mart  ')
    print('  MENU  ')
    print(' ----- ')
    print(' [1] Add New Item  ')
    print(' [2] Show Items ')
    print(' ----- ')
```

```
print(' [3] Exit Program  ')\nprint('-----')
```

```
def main():\n    loop=1\n    while loop:\n        main_menu()\n        ch=get_choice()\n\n        if ch==1:\n            add_new_item()\n        elif ch==2:\n            show_items()\n        elif ch==3:\n            exit_program()\n        else:\n            print('invalid choice...')
```

```
if __name__=='__main__':\n    main()
```

Output:



```
File Example 3\n-----\n\n  City Mart\n  MENU\n\n-----\n\n[1] Add New Item\n[2] Show Items\n\n-----\n\n[3] Exit Program\n\n-----\n\nEnter Choice:
```

## Practice

```
File Practice
-----
    G&G store
    Main MENU
-----
[1] Add New Item
[2] Show Items
[3] Update Price
-----
[4] Exit Program
-----
Enter Choice:
```

Item.py

- Id,name,price

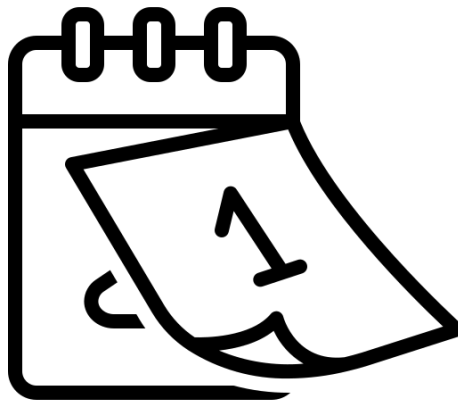
GandGApp.py

---

## *Datetime*

---

🎥 video tutorial 13





# Datetime

To create a date, we can use the `datetime()` class (constructor) of the `datetime` module. The `datetime()` class requires three parameters to create a date: year, month, day.

## Example 1

```
import datetime
```

```
x = datetime.datetime.now()
```

```
print(x)
```

```
print(x.year)
```

```
print(x.month)
```

```
print(x.day)
```

```
print(x.strftime("%Y-%m-%d"))
```

```
print(x.strftime("%d-%m-%Y"))
```

## Output:

```
2024-05-26 01:53:07.712834
```

```
2024
```

```
5
```

```
26
```

```
2024-05-26
```

```
26-05-2024
```

## Example 2:

```
import datetime
```

```
year=2024
```

```
month=5  
day=26  
x = datetime.datetime(year,month,day)  
  
print(x.strftime("%A"))
```

Output:

Sunday

## Practice

Date Format: dd/mm/yyyy

Please Enter Your Birth Date: 02/10/1996

Please Enter The Current Date: 30/06/2023

YOUR Age is: 26 years 9 months 28 days

Hints:

# Calculate days between two dates/Total no. of days

# Check if the given year is a leap year or not.


# If Yes, then Make the total number of days

# in the month of February 29.

---

## *Math & Random*

---

 video tutorial 14



# Math & Random

Python has a set of built-in math functions, including an extensive math module, that allows you to perform mathematical tasks on numbers.

The `random.choice()` function is used in the python string to generate the sequence of characters and digits that can repeat the string in any order.

## Example 1

```
import math
import random

x = min(5, 10, 25)
y = max(5, 10, 25)
#4*4*4
power=pow(4,3)
#squareroot(64) = > 8
sq=math.sqrt(64)
#ceiling no
ceil_no = math.ceil(1.4)
#floor no
floor_no = math.floor(1.4)
#generate random no between 1000 -9999
rand_no=random.randint(1000,9999)

print('minimum value =',x)
print('maxium value =',y)
print('power =',power)
print('square root',sq)
print('ceil no',ceil_no)
print('floor no',floor_no)
print('random no',rand_no)
```

## Output:

```
minimum value = 5
maximum value = 25
power = 64
```

```
square root = 8.0
ceil no = 2
floor no = 1
random no = 5103
```

## Example 2

```
import random
import string
print('Generate Random Password')
random_pw = ''.join([random.choice(string.ascii_letters + string.digits + string.punctuation ) for n in
range(12)])

print('Random Password = ', random_pw)
```

## Output:

```
Generate Random Password
Random Password = +U0'oU'#UbNi
```

## Practice

```
*****
```

```
Special Password Generator
```

```
*****
```

```
Enter Password Length: 8
```

```
Char Set Menu
```

1. Digits
2. Letters
3. Uppercase
4. Generate

```
Enter Menu choice: 1
```

```
Enter Menu choice: 2
```

```
Enter Menu choice: 3
```


```
Enter Menu Choice: 4
```

```
Your Password is : abZ21Mo197
```

---

## *Lambda*

---

 video tutorial 15



# Lambda

A lambda function is an anonymous function. A lambda function can take any number of arguments, but can only have one expression.

## Syntax:

lambda arguments : expression

## Example 1

```
add = lambda a,b : a + b
sub = lambda a,b : a - b
mul = lambda a,b : a * b
div = lambda a,b : a/b
```

```
print(add(5,2))
print(sub(5,2))
print(mul(5,2))
print(div(5,2))
```

Output:

```
7
3
10
2.5
```

## Example 2

```
def to_upper(str):
    return lambda str: str.upper()
```

```
upper=to_upper(str)
print(upper('Welcome to Northern City'))
```

output:

## Output:

WELCOME TO NORTHERN CITY

## Example 3

```
List = [[2,3,4],[1, 4, 16, 64],[3, 6, 9, 12]]
```

```
sortList = lambda x: (sorted(i) for i in x)
largest = lambda x, f : [y[len(y)-1] for y in f(x)]
findLargest = largest(List, sortList)
```

```
print(findLargest)
```

## Output:

[4, 64, 12]

## Practice

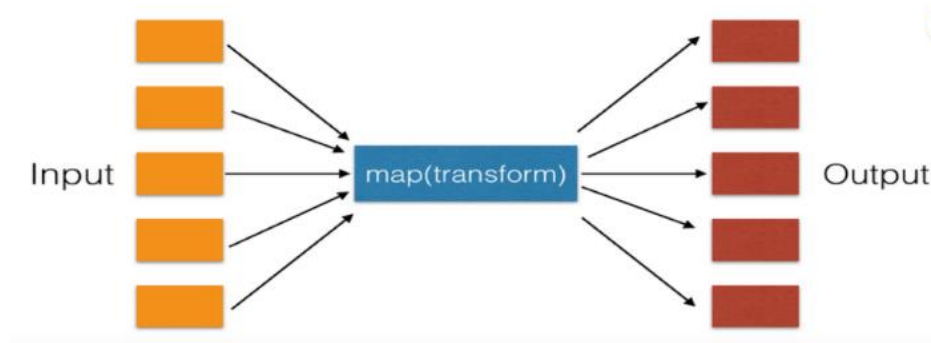
```
List = [[22,33,44],[1, 4, 16, 64],[3, 6, 9, 122]]
```

Output Sample:  
find second largest nos  
[33, 16, 9]



# Map

video tutorial 16



# Map

The `map()` function executes a specified function for each item in an iterable. The item is sent to the function as a parameter.

## Syntax

`map(function, iterables)`

### Example 1

```
def myfunc(a, b):  
    return a + ' '+b  
first_names=["mg mg","bo bo","su myat"]  
last_names=["kyaw","tun","mon"]  
res = map(myfunc, first_names, last_names)
```

```
print(res)
```

```
#convert the map into a list, for readability:  
print(list(res))
```

### Output:

```
<map object at 0x14ec3931f2b0>  
['mg mg kyaw', 'bo bo tun', 'su myat mon']
```

### Example 2

```
def myMapFunc(n):  
    return n.upper()  
  
my_tuple = ('php','java','python','c++','c')
```

```
updated_list = map(myMapFunc, my_tuple)
print(updated_list)
print(list(updated_list))
```

Output:

```
<map object at 0x15094e2c4130>
['PHP', 'JAVA', 'PYTHON', 'C++', 'C']
```

## Example 3:

```
def mapping_course(courses, fees):
    return courses+" : "+fees
```

```
course_titles = ['Computer Basic','Advanced Excel', 'Graphic', 'Video Editing', 'UIUX Design']
course_fees = ('50,000 ks','60,000 ks','100,000 ks','120,000 ks','150,000 ks')
```

```
updated_list = map(mapping_course, course_titles, course_fees)
print(list(updated_list))
```

Output:

```
['Computer Basic : 50,000 ks', 'Advanced Excel : 60,000 ks', 'Graphic : 100,000 ks', 'Video Editing : 120,000 ks', 'UIUX Design : 150,000 ks']
```

## Practice

```
phones = ['iPhone 5','iPhone 6', 'iPhone 7', 'iPhone 8', 'iPhone 10']
release_dates = ['Sep 12,2012','Sep 9,2014','Sep 7,2016','Sep 12,2017','Sep 10,2019']
```

Output:

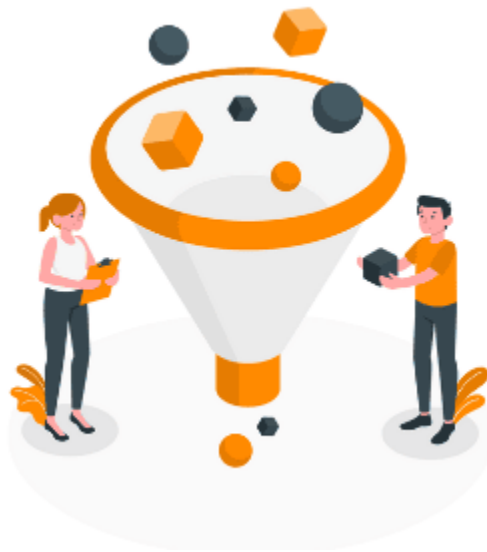
```
{'name': 'iPhone 5', 'release date': 'Sep 12,2012'}, {'name': 'iPhone 6', 'release date': 'Sep 9,2014'},
{'name': 'iPhone 7', 'release date': 'Sep 7,2016'}, {'name': 'iPhone 8', 'release date': 'Sep 12,2017'},
{'name': 'iPhone 10', 'release date': 'Sep 10,2019'}
```

---

## *Filter*

---

🎥 video tutorial 17



# Filter

The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

## Example 1

```
items=[
    {"id":1,"brand":"dell","cpu":"corei3","ram":"8GB","price":400},
    {"id":2,"brand": "acer","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":3,"brand": "hp","cpu": "corei5", "ram": "8GB", "price": 600},
    {"id":4,"brand": "lenovo","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":5,"brand": "dell","cpu": "corei7", "ram": "16GB", "price": 800},
    {"id":6,"brand": "dell","cpu": "corei5", "ram": "16GB", "price": 500},
    {"id":7,"brand": "hp","cpu": "corei3", "ram": "16GB", "price": 700},
    {"id":8,"brand": "hp","cpu": "corei3", "ram": "8GB", "price": 500},
    {"id":9,"brand": "acer","cpu": "corei7", "ram": "32GB", "price": 900},
    {"id":10,"brand": "acer","cpu": "corei3", "ram": "8GB", "price": 500},
    {"id":11,"brand": "asus","cpu": "corei7", "ram": "16GB", "price": 800},
    {"id":12,"brand": "asus","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":13,"brand": "acer","cpu": "corei7", "ram": "32GB", "price":1200},
    {"id":14,"brand": "acer","cpu": "corei7", "ram": "16GB", "price": 500},
    {"id":15,"brand": "lenovo","cpu": "corei3", "ram": "16GB", "price": 500},
    {"id":16,"brand": "acer","cpu": "corei3", "ram": "16GB", "price": 500}
];
```

```
print('display dell laptop computer list')
```

```
dell_laptops=list(filter(lambda item:item['brand']=='dell',items))
```

```
for item in dell_laptops:
    print(item)
```

```
print('display acer laptop computer list')

acer_laptops=list(filter(lambda item:item['brand']=='acer',items))

for item in acer_laptops:
    print(item)

print('display laptop list which price is greater than 600 ')

morethan_600_laptops=list(filter(lambda item:item['price']>=600,items))

for item in morethan_600_laptops:
    print(item)

print('display laptop cpu corei7 list ')

corei7_laptops=list(filter(lambda item:item['cpu']=='corei7',items))

for item in corei7_laptops:
    print(item)
```

## Output:

```
display dell laptop computer list
{'id': 1, 'brand': 'dell', 'cpu': 'corei3', 'ram': '8GB', 'price': 400}
{'id': 5, 'brand': 'dell', 'cpu': 'corei7', 'ram': '16GB', 'price': 800}
{'id': 6, 'brand': 'dell', 'cpu': 'corei5', 'ram': '16GB', 'price': 500}
display acer laptop computer list
{'id': 2, 'brand': 'acer', 'cpu': 'corei3', 'ram': '16GB', 'price': 500}
{'id': 9, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 900}
{'id': 10, 'brand': 'acer', 'cpu': 'corei3', 'ram': '8GB', 'price': 500}
{'id': 13, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 1200}
{'id': 14, 'brand': 'acer', 'cpu': 'corei7', 'ram': '16GB', 'price': 500}
{'id': 16, 'brand': 'acer', 'cpu': 'corei3', 'ram': '16GB', 'price': 500}
display laptop list which price is greater than 600
{'id': 3, 'brand': 'hp', 'cpu': 'corei5', 'ram': '8GB', 'price': 600}
{'id': 5, 'brand': 'dell', 'cpu': 'corei7', 'ram': '16GB', 'price': 800}
{'id': 7, 'brand': 'hp', 'cpu': 'corei3', 'ram': '16GB', 'price': 700}
{'id': 9, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 900}
{'id': 11, 'brand': 'asus', 'cpu': 'corei7', 'ram': '16GB', 'price': 800}
{'id': 13, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 1200}
display laptop cpu corei7 list
{'id': 5, 'brand': 'dell', 'cpu': 'corei7', 'ram': '16GB', 'price': 800}
{'id': 9, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 900}
{'id': 11, 'brand': 'asus', 'cpu': 'corei7', 'ram': '16GB', 'price': 800}
{'id': 13, 'brand': 'acer', 'cpu': 'corei7', 'ram': '32GB', 'price': 1200}
{'id': 14, 'brand': 'acer', 'cpu': 'corei7', 'ram': '16GB', 'price': 500}
```

## Practice

```
-----  
Data Filter Practice  
MENU  
-----  
[1] Show T-Shirt List  
[2] Show Bag List  
[3] Show Hat List  
[4] Show item which price is more than 50 USD  
-----  
Enter Choice:
```

Dataset:


```
item_list=[  
{"id":1,"name":"t-shirt","price":29,"size":"medium"},  
{"id":2,"name":"t-shirt","price":20,"size":"large"},  
{"id":3,"name":"bag","price":90,"size":"medium"},  
{"id":4,"name":"t-shirt","price":19,"size":"small"},  
{"id":5,"name":"hat","price":70,"size":"medium"},  
{"id":6,"name":"bag","price":75,"size":"medium"},  
{"id":7,"name":"t-shirt","price":29,"size":"medium"},  
{"id":8,"name":"hat","price":58,"size":"medium"},  
{"id":9,"name":"t-shirt","price":65,"size":"medium"},  
{"id":10,"name":"t-shirt","price":29,"size":"medium"},  
{"id":11,"name":"bag","price":29,"size":"medium"},  
{"id":12,"name":"bag","price":75,"size":"large"},  
{"id":13,"name":"t-shirt","price":29,"size":"medium"},  
{"id":14,"name":"t-shirt","price":29,"size":"small"},  
{"id":15,"name":"bag","price":65,"size":"small"}  
]
```



---

## *Regular Expression (RegEx)*

---

 video tutorial 18



# Regular Expression

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

## Metacharacters or search keywords

- 
- [] A set of characters
- \ Signals a special sequence (can also be used to escape special characters)
- . Any character (except newline character)
- ^ Starts with
- \$ Ends with
- \* Zero or more occurrences
- + One or more occurrences
- ? Zero or one occurrences
- { } Exactly the specified number of occurrences
- | Either or
- () Capture and group

\A Returns a match if the specified characters are at the beginning of the string     "\AThe"

\b Returns a match where the specified characters are at the beginning or at the end of a word  
(the "r" in the beginning is making sure that the string is being treated as a "raw string")  
r"\bain"

r"ain\b"

\B Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word

(the "r" in the beginning is making sure that the string is being treated as a "raw string")  
r"\Bain"

r"ain\B"

\d Returns a match where the string contains digits (numbers from 0-9)     "\d"

\D Returns a match where the string DOES NOT contain digits     "\D"

\s Returns a match where the string contains a white space character     "\s"

\S Returns a match where the string DOES NOT contain a white space character     "\S"

\w Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character)     "\w"

\W Returns a match where the string DOES NOT contain any word characters     "\W"

`\Z` Returns a match if the specified characters are at the end of the string `"Spain\Z"`

`[arn]` Returns a match where one of the specified characters (a, r, or n) is present

`[a-n]` Returns a match for any lower case character, alphabetically between a and n

`[^arn]` Returns a match for any character EXCEPT a, r, and n

`[0123]` Returns a match where any of the specified digits (0, 1, 2, or 3) are present

`[0-9]` Returns a match for any digit between 0 and 9

`[0-5][0-9]` Returns a match for any two-digit numbers from 00 and 59

`[a-zA-Z]` Returns a match for any character alphabetically between a and z, lower case OR upper case

`[+]` In sets, `+`, `*`, `.`, `|`, `()`, `$`, `{}` has no special meaning, so `[+]` means: return a match for any `+` character in the string

## Example 1

```
import re
```

```
str="hello this is my phone no 09449008972 and my sister's no is 09425029450 and my uncle no is 09425028458"
```

```
phone_list=re.findall('[\d]{11}',str)
```

```
for phone in phone_list:  
    print(phone)
```

Output:

```
09449008972  
09425029450  
09425028458
```

## Example 2

```
import re
```

```
str="hello this is my email northerncity@gmail.com and my sister's email is susu@gmail.com and my  
uncle no is aung77@gmail.com"
```

```
email_list=re.findall('[0-9a-zA-Z\.-]+\@[a-zA-Z0-9\.-]+\.',str)
```

```
for email in email_list:  
    print(email)
```

Output:

```
northerncity@gmail.com  
susu@gmail.com  
aung77@gmail.com
```

Practice

```
Practice1  
-----  
    Data Filter  
    Menu  
-----  
[1] Extract Email  
[2] Phone no  
[3] City Name  
-----  
[4] Exit Program  
-----  
Enter Choice:
```

Dataset:

```
data=(" yangon this is dummy text and email 1 mgmg@gmail.com and his no is 09449008977 and mg  
mg is from Mandalay "
```

```
    "second dummy mandalay text is about susu, she is from Yangon and her phone no is  
09425025778, you can also reach"
```

```
    "her through this no 0977884577 and email susu@gmail.com ")
```

---

## *Numpy*

---

📺 video tutorial 19



# Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

## Example 1

```
import numpy as np

#1-d array
nums=np.array([1,2,3,4,5,6,7,8,9])
print("one dimensional array")
print(nums.shape)

nums=np.array([[1,2,3,4],[5,6,7,8]])
print("2 dimensional array")
print(nums.shape)

nums=np.array([ [1,2,3,4],[5,6,7,8],[9,10,11,12]] , [[21,22,23,24],[25,26,27,28],[29,30,31,32]] )
print("3 dimensional array")
print(nums.shape)
```

## Output:

```
one dimensional array
(9,)
2 dimensional array
(2, 4)
3 dimensional array
(2, 3, 4)
```

## Example 2

### #Shape and Reshape Example

```
import numpy as np
```

```
nums=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newArr=nums.reshape(3,4)
print('Reshape 1 D array to 2 D Array')
print(newArr)
```

```
nums=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newArr=nums.reshape(3,2,2)
print('Reshape 1 D array to 3 D Array')
print(newArr)
```

```
nums=np.array([ [1,2],[3,4]] , [[5,6],[7,8]] , [[9,10],[11,12]] )

newArr=nums.reshape(-1)
print('Reshape 3-D to 1-D array')
print(newArr)
```

```
nums=np.array([ [1,2],[3,4]] , [[5,6],[7,8]] , [[9,10],[11,12]] )

newArr=nums.reshape(2,2,-1)
print('Reshape 3-D to 2-D array')
print(newArr)
```

```
nums=np.array([ [1,2,3,4],[5,6,7,8]] )

newArr=nums.reshape(-1)
print('Reshape 2-D to 1-D array')
print(newArr)
```

## Output:



```
Reshape 1 D array to 2 D Array
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Reshape 1 D array to 3 D Array
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
Reshape 3-D to 1-D array
[ 1  2  3  4  5  6  7  8  9 10 11 12]
Reshape 3-D to 2-D array
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
Reshape 2-D to 1-D array
[1 2 3 4 5 6 7 8]
```

## Example 3

#Indexing

import numpy as np

nums=np.array([41,22,33,44,15,16,17,8,9])

```
newArr=nums[1:3]
print('print nums Index between 1 and 3 ')
print(newArr)

nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index from 0 to 5 ')
newArr=nums[:5]
print(newArr)

nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index -3 to -1 ')
newArr=nums[-3:-1]
print(newArr)

nums=np.array([41,22,33,44,15,16,17,8,9])
print('print nums Index -3 to end ')
newArr=nums[-3:]
print(newArr)
```

## Output:

```
print nums Index between 1 and 3
[22 33]
print nums Index from 0 to 5
[41 22 33 44 15]
print nums Index -3 to -1
[17  8]
print nums Index -3 to end
[17  8  9]
```

## Example 4

```
import numpy as np
```

```
obj_array=[
```

```
{'name':'apple','price':2000},
{'name':'orange','price':1500},
{'name': 'honeydue', 'price': 5000},
{'name': 'lime', 'price': 200},
{'name':'mango','price':1000},
{'name':'banana','price':2500},
{'name': 'watermelon', 'price': 3000},
{'name': 'lychee', 'price': 4500}
]
```

```
items=np.array(obj_array)
filter_arr=[]
for item in items:
    if item['price']>=2000:
        filter_arr.append(True)
    else:
        filter_arr.append(False)

print('Item List item price is greater than 2000 ')
item_list=items[filter_arr]
print(item_list)
```

## Output:

```
Item List item price is greater than 2000
[{'name': 'apple', 'price': 2000} {'name': 'honeydue', 'price': 5000}
{'name': 'banana', 'price': 2500} {'name': 'watermelon', 'price': 3000}
{'name': 'lychee', 'price': 4500}]
```

## Practice

Dataset:

```
item_array=[
```

```
{'name':'tshirt','make':'china','price':30000,'category':'clothes'},  
{'name':'Air Jordan MX','make':'china','price':500000,'category':'shoe'},  
{'name':'Air Max','make':'vietnam','price':100000,'category':'shoe'},  
{'name':'Dress','make':'us','price':30000,'category':'clothes'},  
{'name':'Hat','make':'china','price':30000,'category':'accessories'},  
{'name':'sunglass','make':'china','price':30000,'category':'accessories'},  
{'name':'tshirt','make':'us','price':50000,'category':'clothes'},  
{'name':'watch','make':'china','price':300000,'category':'clothes'},  
{'name':'pant','make':'us','price':30000,'category':'clothes'},  
{'name':'shirt','make':'china','price':55000,'category':'clothes'},  
{'name':'bag','make':'vietnam','price':30000,'category':'accessories'},  
{'name':'tshirt','make':'china','price':10000,'category':'clothes'}  
]
```

## Output:

```
Practice1  
-----  
City Mart  
Menu  
-----  
[1] Show accessories  
[2] China made items  
[3] Price is greater than 100,000  
-----  
[4] Exit Program  
-----  
Enter Choice:
```

---

## *Pandas*

---

🎥 video tutorial 20



# Pandas

Pandas is a Python library. Pandas is used to analyze data.

## Example 1

Data.csv

```
Duration,Date,Pulse,Maxpulse,Calories
60,'2024/02/12',110,130,409.1
60,'2024/02/13',117,145,
600,'2024/02/14',103,135,340
45,'2024/02/15',109,175,282.4
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
45,'2024/02/16',117,148,406
60,20240217,102,127,
60,'2024/02/18',110,136,374
450,'2024/02/19',104,134,253.3
30,'2024/02/20',109,133,195.1
60,'2024/02/21',98,124,269
250,'2024/02/22',103,147,329.3
60,'2024/02/23',100,120,250.7
60,'2024/02/24',106,128,345.3
```

Code:

```
import pandas as pd

#read file
df=pd.read_csv('data.csv')
#show column names and their data types
print(df.dtypes)

#show statistics
print(df.describe())

#show top 5 rows
print(df.head())

#show top 20 rows
print(df.head(20))

#show bottom 5 rows
print(df.tail())

#set max rows
pd.set_option('display.max_rows',500)
print(df)

output:
#print(df.dtypes)
```

```
Duration      int64
Date          object
Pulse         int64
Maxpulse      int64
Calories      float64
dtype: object
```

```
#print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Duration    22 non-null     int64
1   Date        22 non-null     object
2   Pulse       22 non-null     int64
3   Maxpulse    22 non-null     int64
4   Calories    20 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1012.0+ bytes
None
```

```
#print(df.describe())
```

Output:

```
C:\Users\ DELL\PycharmProjects\basic_lessons\.venv\Scripts\python
      Duration      Pulse  Maxpulse  Calories
count  22.000000  22.000000  22.000000  20.000000
mean   102.045455  110.954545  141.545455  355.410000
std    145.452161   6.827656   12.113461   68.721474
min     30.000000   98.000000  120.000000  195.100000
25%     45.000000  104.500000  133.250000  317.575000
50%     45.000000  113.500000  147.500000  406.000000
75%     60.000000  117.000000  148.000000  406.000000
max     600.000000  117.000000  175.000000  409.100000

Process finished with exit code 0
```

```
#display top 5 rows - default
```

```
Print(df.head())
```

Output:

```
Duration      Date  Pulse  Maxpulse  Calories
60  '2024/02/12'   110     130     409.1
60  '2024/02/13'   117     145      NaN
600 '2024/02/14'   103     135     340.0
45  '2024/02/15'   109     175     282.4
45  '2024/02/16'   117     148     406.0
```



```
#display top 20 rows – customize  
Print(df.head(20))
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3

```
#display last row 5 – default  
Print(df.tail())
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

```
#set maximum rows  
pd.set_option('display.max_rows',500)  
print(df)
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

## Example 2

#Removing missing values (NaN)

#command: dropna()

import pandas as pd

#read file

df=pd.read\_csv('data.csv')

pd.set\_option('display.max\_rows',200)

#new\_df=df.dropna()

print(df.to\_string())

Output:

	Duration	Date	Pulse	Maxpulse	Calories
	60	'2024/02/12'	110	130	409.1
	60	'2024/02/13'	117	145	NaN
	600	'2024/02/14'	103	135	340.0
	45	'2024/02/15'	109	175	282.4
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
	45	'2024/02/16'	117	148	406.0
0	45	'2024/02/16'	117	148	406.0
1	45	'2024/02/16'	117	148	406.0
2	45	'2024/02/16'	117	148	406.0
3	45	'2024/02/16'	117	148	406.0
4	60	20240217	102	127	NaN
5	60	'2024/02/18'	110	136	374.0
6	450	'2024/02/19'	104	134	253.3
7	30	'2024/02/20'	109	133	195.1
8	60	'2024/02/21'	98	124	269.0
9	250	'2024/02/22'	103	147	329.3
0	60	'2024/02/23'	100	120	250.7
1	60	'2024/02/24'	106	128	345.3

#After removing missing values

```
import pandas as pd
```

```
#read file
```

```
df=pd.read_csv('data.csv')
```

```
pd.set_option('display.max_rows',200)
```

```
new_df=df.dropna()
```

```
print(new_df.to_string())
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

### Example 3:

```
#Original Data Set  
import pandas as pd
```

```
df = pd.read_csv('data.csv')  
print(df)
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	45	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	45	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

#After Removing Duplicated Data

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
# remove opeation
```

```
df.drop_duplicates(inplace=True)
```

```
#after removing duplicates
```

```
print(df)
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1
18	60	'2024/02/21'	98	124	269.0
19	250	'2024/02/22'	103	147	329.3
20	60	'2024/02/23'	100	120	250.7
21	60	'2024/02/24'	106	128	345.3

## Example 4:

#Original Data with Wrong Data

```
C:\Users\Dell\PycharmProjects\basic_lessons\.venv\Scripts\python.exe C:\Users\Dell\Py
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	600	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	545	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	300	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	450	'2024/02/19'	104	134	253.3
17	30	'2024/02/20'	109	133	195.1

#After Correcting Wrong Data

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
```

```
for x in df.index:
```

```
    if df.loc[x,'Duration']>60:
```

```
        df.loc[x,'Duration']=60
```

```
print(df)
```

Output:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2024/02/12'	110	130	409.1
1	60	'2024/02/13'	117	145	NaN
2	60	'2024/02/14'	103	135	340.0
3	45	'2024/02/15'	109	175	282.4
4	45	'2024/02/16'	117	148	406.0
5	60	'2024/02/16'	117	148	406.0
6	45	'2024/02/16'	117	148	406.0
7	60	'2024/02/16'	117	148	406.0
8	45	'2024/02/16'	117	148	406.0
9	45	'2024/02/16'	117	148	406.0
10	45	'2024/02/16'	117	148	406.0
11	45	'2024/02/16'	117	148	406.0
12	45	'2024/02/16'	117	148	406.0
13	45	'2024/02/16'	117	148	406.0
14	60	20240217	102	127	NaN
15	60	'2024/02/18'	110	136	374.0
16	60	'2024/02/19'	104	134	253.3

## Example 5:

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.info())
```

## Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  -
 0   Duration    22 non-null    int64  
 1   Date        22 non-null    object  
 2   Pulse       22 non-null    int64  
 3   Maxpulse    22 non-null    int64  
 4   Calories    20 non-null    float64 
dtypes: float64(1), int64(3), object(1)
memory usage: 1012.0+ bytes
None
```

#After Converting Date Data type to Date Time Data type



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Duration    22 non-null    int64
1   Date        21 non-null    datetime64[ns]
2   Pulse       22 non-null    int64
3   Maxpulse    22 non-null    int64
4   Calories    20 non-null    float64
dtypes: datetime64[ns](1), float64(1), int64(3)
memory usage: 1012.0 bytes
None

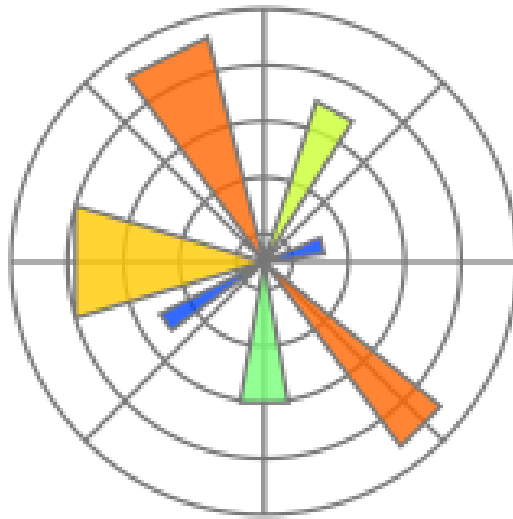
Process finished with exit code 0
```

---

# Matplotlib

---

📺 video tutorial 21



# Matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely.

## Example 1

```
import matplotlib.pyplot as plt
import numpy as np

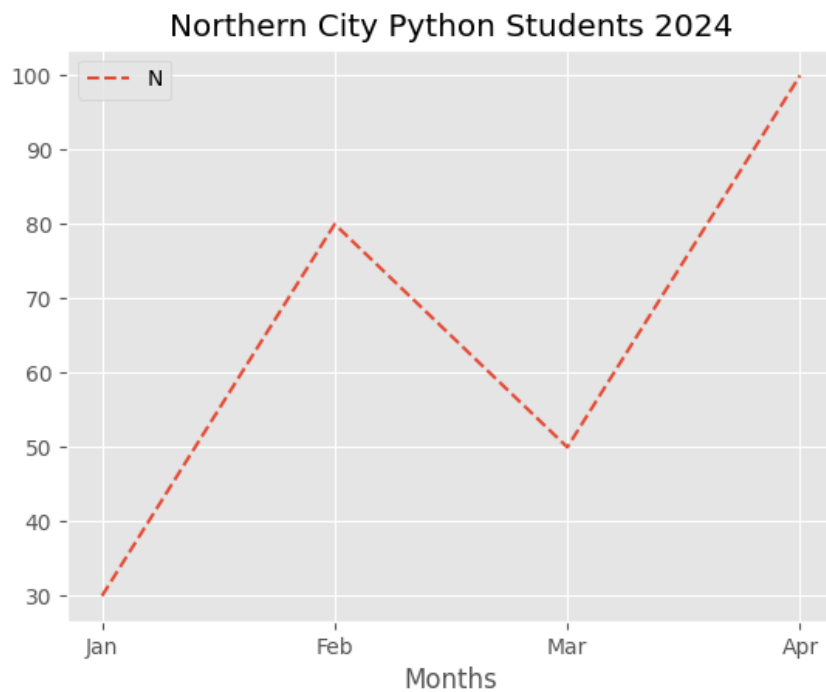
#print(plt.style.available)
plt.style.use('ggplot')
months = ['Jan','Feb','Mar','Apr']
qty = [30, 80, 50, 100]

plt.plot(months,qty, linestyle = 'dashed')
plt.title('Northern City Python Students 2024')

plt.xlabel("Months")
plt.legend("No of Students")
#plt.savefig('line_graph.png')
plt.show()
```

Output:

Figure 1



## Example 2

#Myanmar Cities and their population

```
from matplotlib import pyplot as plt
```

```
dev_x=['Yangon','Mandalay','Mawlamyine','Taunggyi','Lashio','Myitkyina']  
dev_y=[6200000,2500000,1600000,550000,210000,870000];
```

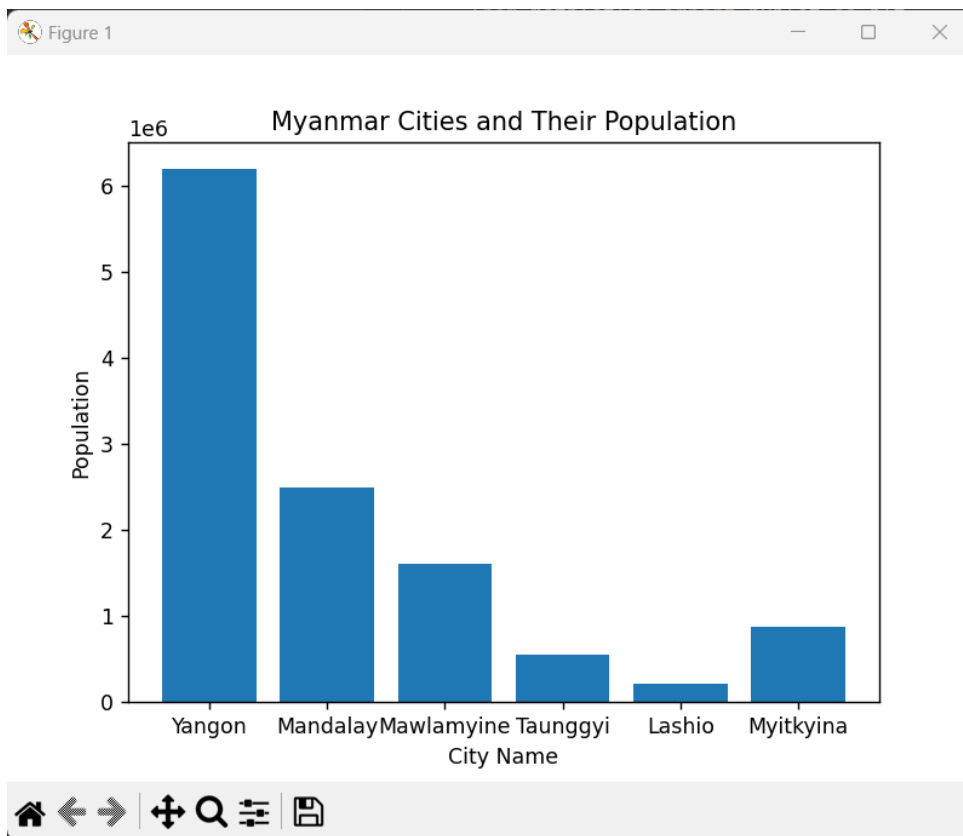
```
#setting bar graph  
plt.bar(dev_x,dev_y)
```

```
#describing labels of the graph  
plt.xlabel('City Name')  
plt.ylabel('Population')
```

```
#describing graph title
plt.title('Myanmar Cities and Their Population')

#show graph
plt.show()
```

## Output:



## Example 3

```
#2024 Northern City entrolled Programming Student
#According to Months and Populations
import numpy as np
from matplotlib import pyplot as plt
```

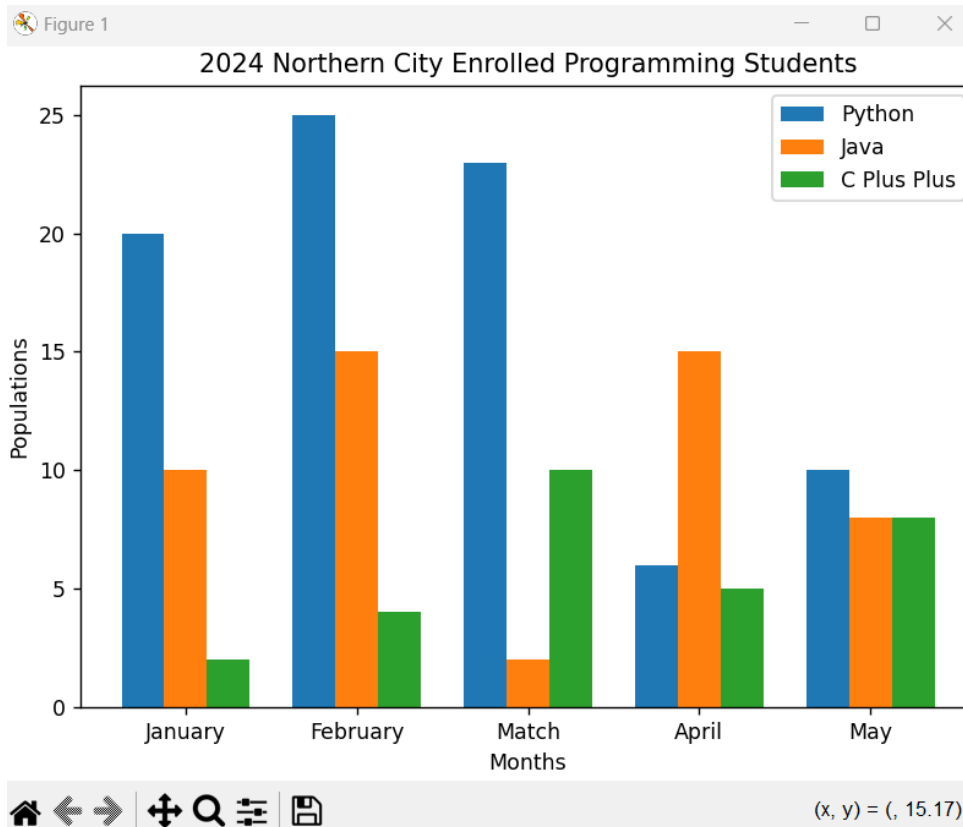
```
# Setting the width of the bars
bar_width = 0.25
x=np.arange(5)
# Calculating bar positions for both groups
bar_positions1 = np.arange(5)
bar_positions2 = bar_positions1 + bar_width
bar_positions3 = bar_positions2 + bar_width

ax = plt.subplots(layout='constrained')
dev_x=('January','February','March','April','May')
dev_y_python=[20,25,23,6,10];
plt.bar(bar_positions1,dev_y_python,width=bar_width,label='Python')
dev_y_java=[10,15,2,15,8];
plt.bar(bar_positions2,dev_y_java,width=bar_width,label='Java')
dev_y_cpp=[2,4,10,5,8];
plt.bar(bar_positions3,dev_y_cpp,width=bar_width,label='C Plus Plus')

plt.legend(['Python','Java','C Plus Plus'])
plt.xlabel('Months')
## plt.xticks(range(len(list)),col_names)
plt.xticks(x+bar_width,dev_x)
plt.ylabel('Populations')
plt.title('2024 Northern City Enrolled Programming Students')

plt.show()
```

## Output:



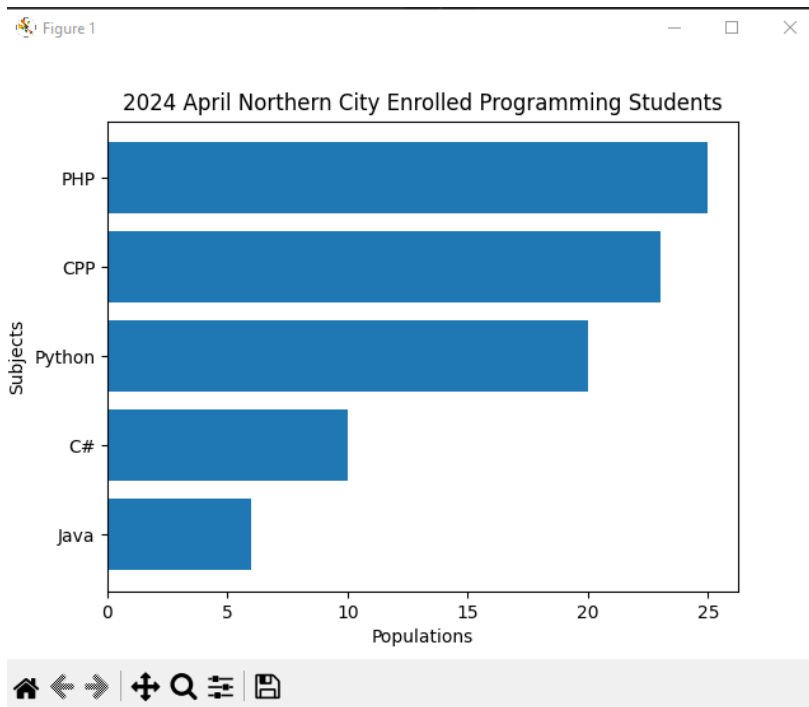
## Example 4

```
#2024 April Entrolled Student List
import numpy as np
from matplotlib import pyplot as plt

dev_x=('Java','C#','Python','CPP','PHP')
dev_y=[20,25,23,6,10]
#sorted_y=dev_y.sort(reverse=True)
sorted_y=dev_y.sort()
plt.barh(dev_x,dev_y,data=sorted_y)
plt.ylabel('Subjects')
plt.xlabel('Populations')
plt.title('2024 April Northern City Enrolled Programming Students')

plt.show()
```

## Output:



## Example 5

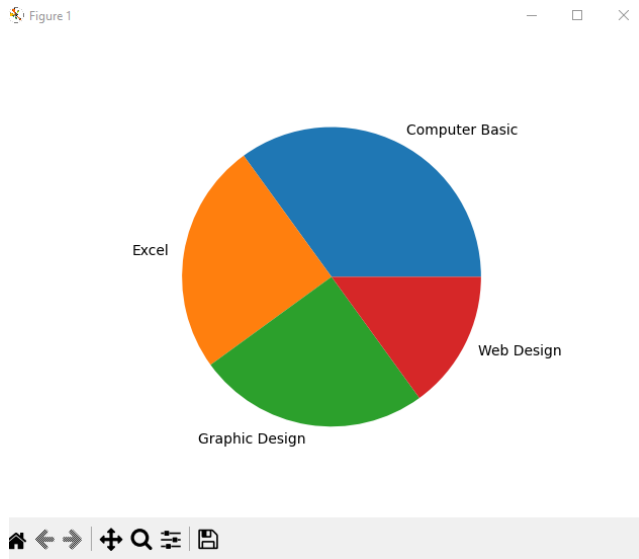
```
import matplotlib.pyplot as plt
import numpy as np
```

```
data = np.array([35, 25, 25, 15])
mylabels = ["Computer Basic", "Excel", "Graphic Design", "Web Design"]
```

```
plt.pie(data, labels = mylabels)
plt.show()
```

Ouput:





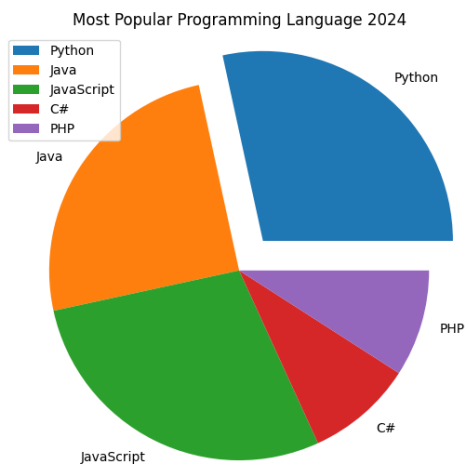
## Example 6

```
import matplotlib.pyplot as plt
import numpy as np

data = np.array([25, 22, 25, 8,8])
mylabels = ["Python", "Java", "JavaScript", "C#","PHP"]
myexplode = [0.2, 0, 0, 0,0]

plt.pie(data, labels = mylabels,explode=myexplode)
plt.title("Most Popular Programming Language 2024")
plt.legend(["Python", "Java", "JavaScript", "C#","PHP"])
plt.show()
```

Output:



# Practice

Title: Northern City Enrolled Students 2023

Dataset:

```
data = [180, 420, 250, 210, 80]
```

```
subjects = ["Python", "Java", "JavaScript", "C#", "PHP"]
```

## Matplotlib Practice

### Main MENU

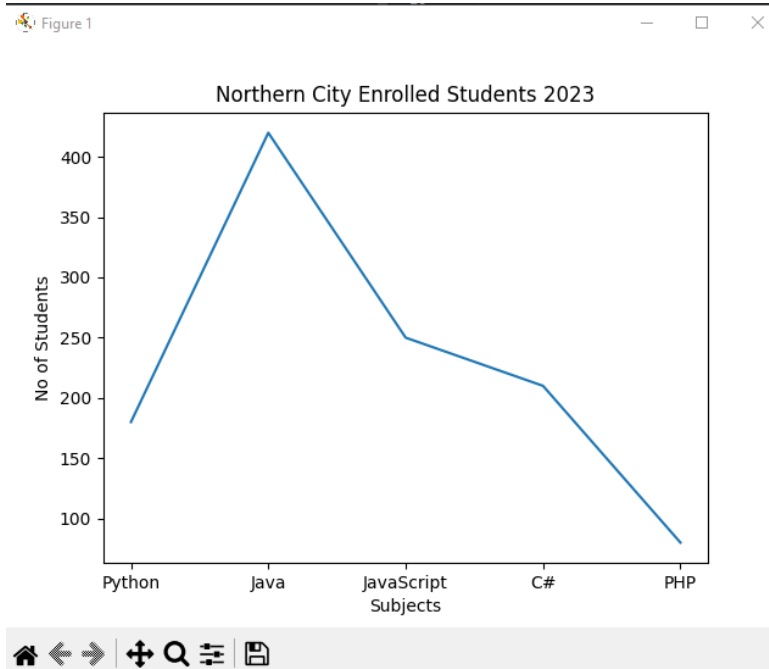
- [1] Draw Line Graph
- [2] Draw Vertical Bar Graph
- [3] Draw Horizontal Bar Graph
- [4] Draw Pie Graph

- [5] Exit Program

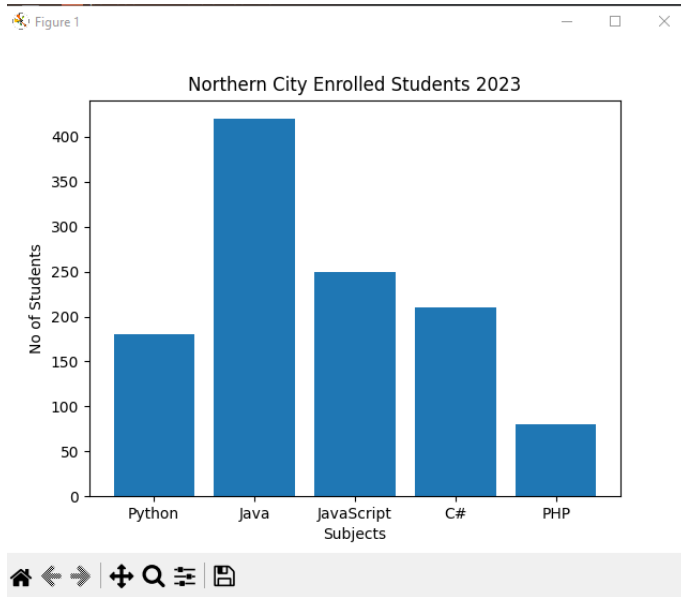
Enter choice:

## Output:

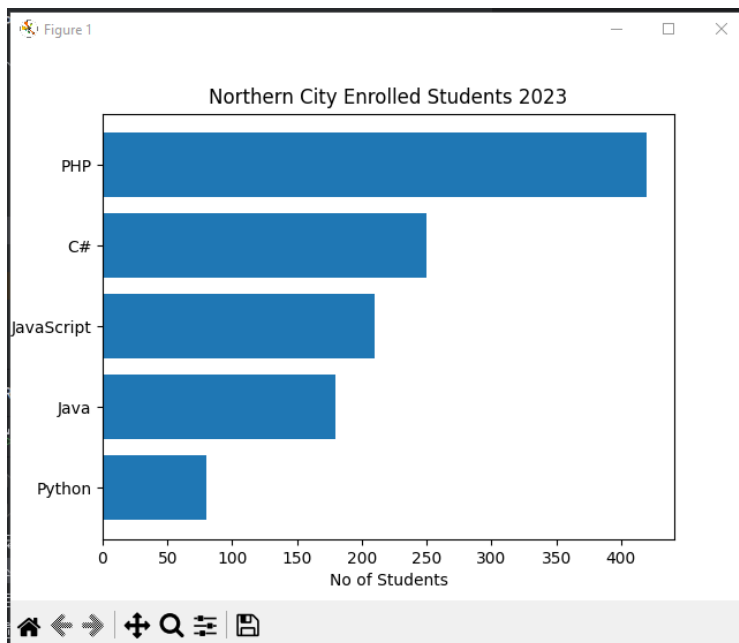
Option-1 Output:



## Option-2 Output:



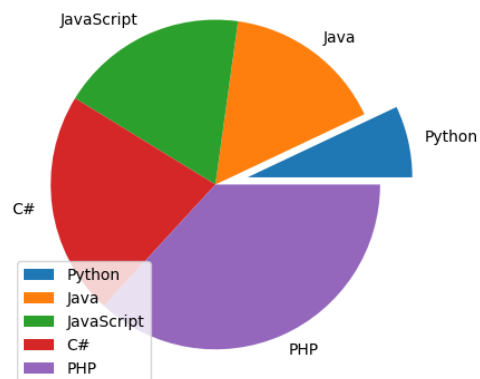
## Option-3 Output:



## Option 4 – Output:

Figure 1

Northern City Enrolled Students 2023



## References:

- <https://www.w3schools.com/>
- <https://www.sololearn.com/en/>
- <https://www.freecodecamp.org/>