Name	**SAVITHA S**
Date	**05 September 2022**
Team ID	**PNT2022TMID41467**
Project Name	**AI-Powered Nutrition Analyzer for Fitness Enthusiasts**

Python

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

```
Hello World
```

```
print('Hello World')
```

```
Hello World
```

```
1+2
```

```
3
```

```
print(1+2)
```

```
3
```

```
1/2
```

```
0.5
```

```
5*5
```

```
25
```

Keywords

```
import keyword

print(keyword.kwlist)
```

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

Identifier

```
anna_unv = 5
```

```
anna_unv
```

```
5
```

```
assert = 5
```

```
  File "<ipython-input-12-afa64a788a4a>", line 1
    assert = 5
           ^


SyntaxError: invalid syntax
```

```
1anna_unv = 5
```

```
  File "<ipython-input-13-9b81eb200c00>", line 1
    1anna_unv = 5
             ^
SyntaxError: invalid syntax
```

```
anna_unv_1 = 5
```

```
anna_unv1 = 5
```

```
anna_unv@ = 5
```

```
  File "<ipython-input-16-e0f83a281de9>", line 1
    anna_unv@ = 5
              ^
SyntaxError: invalid syntax
```

Comments

```
# Age is stored in variable 'a'
```

```
a = 10
```

```
'''Age is stored in
variable a'''
```

```
a = 10
```

```
"""Age is stored in
variable a"""
```

```
a = 10
```

Statements

 # Single line statement

```
a = 10+
3

print(a)
```

```
  File "<ipython-input-24-097d3e8c692a>", line 2
    a = 10+
          ^
SyntaxError: invalid syntax
```

 # multi line statement

```
a = 10+3\
+24

print(a)
```

```
37
```

Variable assignment

```
int_var = 6

int_var
```

```
6
```

```
float_var = 6.12
float_var
```

```
6.12
```

```
str_var = 'Hari'
str_var
```

```
{"type":"string"}
```

```
a,b,c = 6,6.12,'Hari'

c
```

```
{"type":"string"}
```

Data Type

1. Numeric Type

```
num = 10
num
```

```
type(num)
```

```
int
```

```
num = 10.12
num
```

```
10.12
```

```
type(num)
```

```
float
```

```
num = 10+20j
num
```

```
(10+20j)
```

```
type(num)
```

```
complex
```

2. Sequence Type

String

```
str1 = "Welcome to AI"
```

```
str1
```

```
{"type":"string"}
```

```
type(str1)
```

```
str
```

String indexing

```
"""
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12
```

```
W e l c o m e        t o       A I
```

```
                -5-4 -3 -2-1
```

```
"""
```

```
{"type":"string"}
```

```
str1[7]
```

```
{"type":"string"}
```

```
str1[-4]
```

```
{"type":"string"}
```

String Slicing

```
str1[0:7]
```

```
{"type":"string"}
```

```
str1[3:7]
```

```
{"type":"string"}
```

```
str1
```

```
{"type":"string"}
```

```
str1[-5:-3]
```

```
{"type":"string"}
```

```
a = "Anna University"
a
```

```
{"type":"string"}
```

```
a[0:4]
```

```
{"type":"string"}
```

```
a[-15:-9]
```

```
{"type":"string"}
```

String Concatenation

```
fname = 'Hari'
lname = 'Prabu'
```

```
fname+lname
```

```
{"type":"string"}
```

```
fname,lname
```

```
('Hari', 'Prabu')
```

```
print(fname+' '+lname)
```

```
Hari Prabu
```

String function

```
str1 = "   Hello Students   "
str1
```

```
{"type":"string"}
```

```
str1.strip()
```

```
{"type":"string"}
```

```
str1.lstrip()
```

```
{"type":"string"}
```

```
str1.rstrip()
```

```
{"type":"string"}
```

```
# HELLO, Hello, hello
```

```
str1
```

```
{"type":"string"}
```

```
str1.lower()
```

```
{"type":"string"}
```

```
str1.upper()
```

```
{"type":"string"}
```

```
str1.replace('Hello','Welcome')
```

```
{"type":"string"}
```

```
str1.replace(' ','')
```

```
{"type":"string"}
```

```
str1.split()
```

```
['Hello', 'Students']
```

List

```
list1 = []
```

```
list1
```

```
[]
```

```
type(list1)
```

```
list
```

```
list1 = [1,2,3]
list1
```

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

```
type(list1)
```

```
list
```

```
list1 = [1.11,2.12,3.75]
list1
```

```
[1.11, 2.12, 3.75]
```

```
list1 = ['AI','DL','ML']
list1
```

```
['AI', 'DL', 'ML']
```

```
list1 = [1,2.12,'Hari']
list1
```

```
[1, 2.12, 'Hari']
```

```
# List indexing
```

```
list1[2]
```

```
{"type":"string"}
```

```
list1[-1]
```

{"type":"string"}# List

Slicing list1[1:]

```
[2.12, 'Hari']
```

```
list1[-2:]
```

```
[2.12, 'Hari']
```

```
list1[1:3]
```

```
[2.12, 'Hari']
```

```
list1 = [1,2.12,['Hari','Prabu']]
list1
```

```
[1, 2.12, ['Hari', 'Prabu']]
```

```
list1[2][1]
```

```
{"type":"string"}
```

```
list1[-2:]
```

```
[2.12, ['Hari', 'Prabu']]
```

```
list1
[1, 2.12, ['Hari', 'Prabu']]
list1.append('Srikanth')
list1
[1, 2.12, ['Hari', 'Prabu'], 'Srikanth']
list1.insert(1,'Hello')
list1
[1, 'Hello', 2.12, ['Hari', 'Prabu'], 'Srikanth']
list1.remove('Hello')
list1
[1, 2.12, ['Hari', 'Prabu'], 'Srikanth']
list1.remove(2.12)
list1
[1, ['Hari', 'Prabu'], 'Srikanth']
l2 = [1,2,3]
list1+l2
[1, ['Hari', 'Prabu'], 'Srikanth', 1, 2, 3]
list1 = [25,45,1,32,12,11]
list1
[25, 45, 1, 32, 12, 11]
list1.sort()
list1
[1, 11, 12, 25, 32, 45]
list1.sort(reverse=True)
list1
[45, 32, 25, 12, 11, 1]
sorted(list1)
[1, 11, 12, 25, 32, 45]
```

```
list1
```

```
[45, 32, 25, 12, 11, 1]
```

```
list1[2]='Hi'
```

```
list1
```

```
[45, 32, 'Hi', 12, 11, 1]
```

Tuples

```
tup1 = ()
tup1
```

```
()
```

```
type(tup1)
```

```
tuple
```

```
tup1 = (1,2,'Hi','Hey',15.28)
tup1
```

```
(1, 2, 'Hi', 'Hey', 15.28)
```

```
tup1[1]
```

```
2
```

```
tup1[-1]
```

```
15.28
```

```
tup1[1:3]
```

```
(2, 'Hi')
```

```
tup1[-4:-2]
```

```
(2, 'Hi')
```

```
tup1[-1]=21
```

```
---------------------------------------------------------------------------
------
TypeError                                 Traceback (most recent call
last)
<ipython-input-112-c585bc529734> in <module>
----> 1 tup1[-1]=21

TypeError: 'tuple' object does not support item assignment
```

Set

```
set1 = {1,2,3}

type(set1)

set

set1={'Hari','IBM','Hyd','Hyd'}
set1

{'Hari', 'Hyd', 'IBM'}

set1[1]

---------------------------------------------------------------------------
------
TypeError                                 Traceback (most recent call
last)
<ipython-input-120-d27c0eee2e56> in <module>
----> 1 set1[1]

TypeError: 'set' object is not subscriptable

set1

{'Hari', 'Hyd', 'IBM'}

set1.add('Kovai','Chennai')

---------------------------------------------------------------------------
------
TypeError                                 Traceback (most recent call
last)
<ipython-input-124-18cbef330b02> in <module>
----> 1 set1.add('Kovai','Chennai')

TypeError: add() takes exactly one argument (2 given)

set1

{'Hari', 'Hyd', 'IBM', 'Kovai'}

set1.update(['Chennai','TN'])

set1

{'Chennai', 'Hari', 'Hyd', 'IBM', 'Kovai', 'TN'}

set1.remove('TN')

set1

{'Chennai', 'Hari', 'Hyd', 'IBM', 'Kovai'}
```

```
set1.clear()
```

```
set1
```

set() **del**

set1set1

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-132-d18f5a84d934> in <module>
----> 1 set1

NameError: name 'set1' is not defined
```

Dictionary

```
dict1 = {}
```

```
type(dict1)
```

```
dict
```

```
dict1 = {'Name':'Hari','Org':'IBM','Loc':'cbe'}
dict1
```

```
{'Name': 'Hari', 'Org': 'IBM', 'Loc': 'cbe'}
```

```
dict1.keys()
```

```
dict_keys(['Name', 'Org', 'Loc'])
```

```
dict1.values()
```

```
dict_values(['Hari', 'IBM', 'cbe'])
```

```
dict1.items()
```

```
dict_items([('Name', 'Hari'), ('Org', 'IBM'), ('Loc', 'cbe')])
```

```
dict1 = {'Name': ['Hari','Srikanth'], 'Org': 'IBM', 'Loc': 'cbe'}
```

```
dict1
```

```
{'Name': ['Hari', 'Srikanth'], 'Org': 'IBM', 'Loc': 'cbe'}
```

```
dict1['Name'][1]
```

```
{"type":"string"}
```

```
dict1.pop('Loc')
```

```
{"type":"string"}
```

```
dict1
```

```
{'Name': ['Hari', 'Srikanth'], 'Org': 'IBM'}
```

```
dict1.clear()
```

```
dict1
```

```
{}
```

```
del dict1
```

```
dict1
```

```
---------------------------------------------------------------------------

------
NameError                                 Traceback (most recent call
last)
<ipython-input-152-e36219336d90> in <module>
----> 1 dict1

NameError: name 'dict1' is not defined
```

Statements/Condition

```
num = input('Enter int values ')
num
```

```
Enter int values 45.214
```

```
{"type":"string"}
```

```
type(num)
```

```
str
```

```
num = int(input('Enter int values '))
num
```

```
Enter int values 21
```

```
21
```

```
type(num)
```

```
int
```

```
# If statement


num = int(input('Enter int values '))
```

```
if num%2 == 0:
  print("Number is even")
```

```
Enter int values 3
```

```
# If & else statement


num = int(input('Enter int values '))

if num%2 == 0:
  print("Number is even")
else:
  print("Number is odd")
```

```
Enter int values 21
Number is odd
```

```
# If, elif & else statement


x = int(input('Enter int values X = '))
y = int(input('Enter int values Y = '))

if x>y:

  print("X is greater than Y")
elif x==y:

  print("X is equal to Y")
else:
  print("X is lesser than Y")
```

```
Enter int values X = 12
Enter int values Y = 40
X is lesser than Y
```

```
# If, elif & else statement


x = int(input('Enter int values X = '))
y = int(input('Enter int values Y = '))

if x>y:

  print("{} is greater than {}".format(x,y))
elif x==y:

  print("{} is equal to {}".format(x,y))
else:
  print("{} is lesser than {}".format(x,y))
```

```
Enter int values X = 21
Enter int values Y = 12
21 is greater than 12
```

```
My name is Hari
```

```
print("My name is Srikanth")
```

```
My name is Srikanth
```

```
name = input("Enter your name: ")
print("My name is {}".format(name))
```

```
Enter your name: Hari
My name is Hari
```

For loop

```
range(10)
```

```
range(0, 10)
```

```
for i in range(10):
  print(i,end=" ")
```

```
0  1  2  3  4  5  6  7  8  9
```

```
for i in range(10):print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for i in 'Hari Prabu':print(i)
```

```
H
a
r
i

P
r
a
b
u
```

```
for i in enumerate('Hari Prabu'):
  print(i)
```

```
(0, 'H')
(1, 'a')
(2, 'r')
(3, 'i')
(4, ' ')
(5, 'P')
(6, 'r')
(7, 'a')
(8, 'b')
(9, 'u')
```

While loop

```
i = 0
while i<7: print(i)
   i+=1 # i=i+1
```

```
0
1
2
3
4
5
6
```

```
i = 0
while       i<7:
   print(i)i=i+1
```

```
0
1
2
3
4
5
6
```

Functions

```
def func():
  print('Hari')

func()
```

```
Hari
```

```
def user_details(name,userid,country):
  print('Name: ',name)
  print('UserID: ',userid)
  print('Country: ',country)

user_details('Hari',1234,'India')
```

```
Name:  Hari
UserID:  1234
Country:  India
```

```
user_details('Srikanth',1235,'India')
```

```
Name:  Srikanth
UserID:  1235
Country:  India
```

Lambda

```
# Lambda arg:exp
```

```
add = lambda num:num+10add()
```

```
------
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call
last)
<ipython-input-9-d5d29de3ed94> in <module>
----> 1 add()

TypeError: <lambda>() missing 1 required positional argument: 'num'
```

```
add(5)
```

```
15
```

```
mul = lambda x,y,z:x*y*z
```

```
mul(12.56,45.278,2)
```

```
1137.38336
```

```
n = lambda name:name+' Prabu'
```

```
n('Hari')
```

```
{"type":"string"}
```

Numpy

```
import numpy as np
```

```
list1 = [2,5,6]
arr = np.array(list1)

list1,arr

([2, 5, 6], array([2, 5, 6]))

arr[1]

5

dir(arr)

['T',
 '__abs__',
 '__add__',
 '__and__',
 '__array__',
 '__array_finalize__',
 '__array_function__',
 '__array_interface__',
 '__array_prepare__',
 '__array_priority__',
 '__array_struct__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__complex__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__float__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__iand__',
 '__ifloordiv__',
 '__ilshift__',
 '__imatmul__',
```

```
'__imod__',
'__imul__',
'__index__',
'__init__',
'__init_subclass__',
'__int__',
'__invert__',
'__ior__',
'__ipow__',
'__irshift__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lshift__',
'__lt__',
'__matmul__',
'__mod__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rlshift__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__rpow__',
'__rrshift__',
'__rshift__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
```

```
'__sub__',
'__subclasshook__',
'__truediv__',
'__xor__',
'all',
'any',
'argmax',
'argmin',
'argpartition',
'argsort',
'astype',
'base',
'byteswap',
'choose',
'clip',
'compress',
'conj',
'conjugate',
'copy',
'ctypes',
'cumprod',
'cumsum',
'data',
'diagonal',
'dot',
'dtype',
'dump',
'dumps',
'fill',
'flags',
'flat',
'flatten',
'getfield',
'imag',
'item',
'itemset',
'itemsize',
'max',
'mean',
'min',
'nbytes',
'ndim',
'newbyteorder',
'nonzero',
'partition',
'prod',
'ptp',
'put',
'ravel',
'real',
```

```
 'repeat',
 'reshape',
 'resize',
 'round',
 'searchsorted',
 'setfield',
 'setflags',
 'shape',
 'size',
 'sort',
 'squeeze',
 'std',
 'strides',
 'sum',
 'swapaxes',
 'take',
 'tobytes',
 'tofile',
 'tolist',
 'tostring',
 'trace',
 'transpose',
 'var',
 'view']
```

```
type(arr)
```

```
numpy.ndarray
```

```
arr.ndim
```

```
1
```

```
arr
```

```
array([2, 5, 6])
```

```
a = np.array([[2,4,5],[7,8,9]])
a
```

```
array([[2, 4, 5],
       [7, 8, 9]])
```

```
a.ndim
```

```
2
```

```
# arange
```

```
np.arange(10)
```

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

```
range(10)

range(0, 10)

for i in range(10):print(i)

0
1
2
3
4
5
6
7
8
9

a = np.arange(5,11)
a

array([ 5, 6, 7, 8, 9, 10])

a[1:3]

array([6, 7])

np.arange(5,11,3)

array([5, 8])

np.arange(0,50,7)

array([ 0,  7, 14, 21, 28, 35, 42, 49])

# Zeros and ones

np.zeros(3,dtype='int')array([0, 0,

0]) np.zeros(3)


array([0., 0., 0.])

a= np.zeros((4,4))

a.ndim

2

a

array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
```

```
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

```
np.ones((4,4),dtype='int')*15
```

```
array([[15, 15, 15, 15],
       [15, 15, 15, 15],
       [15, 15, 15, 15],
       [15, 15, 15, 15]])
```

 # Line space

```
np.linspace(0,10,5)
```

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

 # Argmax & argmin

```
a = np.array([15,2,17,86,1])
a
```

```
array([15,  2, 17, 86,  1])
```

```
a.argmax()
```

```
3
```

```
a.argmin()
```

```
4
```

```
a[-1]
```

```
1
```

```
a[1:3]
```

```
array([ 2, 17])
```

```
a[-3:-1]
```

```
array([17, 86])
```

```
a.max(),a.min()
```

```
(86, 1)
```

```
a
```

```
array([15, 2, 17, 86,  1])
```

```
a[3]=24
```

```
a[5]=11
```

```
---------------------------------------------------------------------------
------
IndexError                                Traceback (most recent call
last)
<ipython-input-71-1ff844d9e8bc> in <module>
----> 1 a[5]=11

IndexError: index 5 is out of bounds for axis 0 with size 5
```

 # Reshape

```
a= np.array([15,  2, 17, 86,  1, 2])
a
```

```
array([15,  2, 17, 86,  1,  2])
```

```
a.reshape(3,2)
```

```
array([[15,  2],
       [17, 86],
       [ 1,  2]])
```

 # Creating random values

```
np.random.rand(2)
```

```
array([0.83648364, 0.43309337])
```

```
np.random.rand(2,5)
```

```
array([[0.623721  , 0.51200912, 0.36777632, 0.39831869, 0.52220845],
       [0.87711185, 0.457721  , 0.84865154, 0.25295241, 0.81176368]])
```

```
np.random.randn(2,5)
```

```
array([[ 1.82202334, -0.63501578, -1.83936146, -0.34769198,
1.75778546],
       [ 0.45184492,  1.17532952, -0.67522199,  0.07392674,
1.63503973]])
```

```
np.random.randint(2,5)
```

```
4
```

```
a = np.random.randint(2,5,6)
a
```

```
array([2, 2, 4, 3, 2, 3])
```

```
np.sqrt(a)
```

```
array([1.41421356, 1.41421356, 2.        , 1.73205081, 1.41421356,
       1.73205081])
```

```
np.log(a)
```

```
array([0.69314718, 0.69314718, 1.38629436, 1.09861229, 0.69314718,
       1.09861229])
```

```
sorted(a)
```

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

Pandas

```
import pandas as pd
```

```
np.random.rand(5)
```

```
array([0.89801474, 0.62738781, 0.28027938, 0.54479857, 0.31833287])
```

```
dir(s)
```

```
['T',
 '_AXIS_LEN',
 '_AXIS_ORDERS',
 '_AXIS_REVERSED',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__finalize__',
 '__float__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattr__',
```

```
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__int__',
'__invert__',
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__long__',
'__lt__',
'__matmul__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
```

```
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
'_accessors',
'_accum_func',
'_add_numeric_operations',
'_agg_by_level',
'_agg_examples_doc',
'_agg_see_also_doc',
'_align_frame',
'_align_series',
'_arith_method',
'_as_manager',
'_attrs',
'_binop',
'_can_hold_na',
'_check_inplace_and_allows_duplicate_labels',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_axes_from_arguments',
'_construct_result',
'_constructor',
'_constructor_expanddim',
'_convert',
'_convert_dtypes',
'_data',
'_dir_additions',
'_dir_deletions',
'_drop_axis',
'_drop_labels_or_levels',
'_duplicated',
'_find_valid_index',
'_flags',
```

```
'_from_mgr',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cacher',
'_get_cleaned_column_resolvers',
'_get_index_resolvers',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_value',
'_get_values',
'_get_values_tuple',
'_get_with',
'_gotitem',
'_hidden_attrs',
'_index',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_init_dict',
'_init_mgr',
'_inplace_method',
'_internal_names',
'_internal_names_set',
'_is_cached',
'_is_copy',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_item_cache',
'_ixs',
'_logical_func',
'_logical_method',
'_map_values',
'_maybe_update_cacher',
'_memory_usage',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_name',
'_needs_reindex_multi',
'_protect_consolidate',
'_reduce',
'_reindex_axes',
```

```
'_reindex_indexer',
'_reindex_multi',
'_reindex_with_indexers',
'_replace_single',
'_repr_data_resource_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_set_as_cached',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
'_set_labels',
'_set_name',
'_set_value',
'_set_values',
'_set_with',
'_set_with_engine',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_stat_function',
'_stat_function_ddof',
'_take_with_is_copy',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'append',
'apply',
'argmax',
'argmin',
'argsort',
'array',
'asfreq',
'asof',
'astype',
'at',
```

```
'at_time',
'attrs',
'autocorr',
'axes',
'backfill',
'between',
'between_time',
'bfill',
'bool',
'clip',
'combine',
'combine_first',
'compare',
'convert_dtypes',
'copy',
'corr',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'divmod',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtype',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'ewm',
'expanding',
'explode',
'factorize',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
'floordiv',
'ge',
```

```
'get',
'groupby',
'gt',
'hasnans',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
'index',
'infer_objects',
'interpolate',
'is_monotonic',
'is_monotonic_decreasing',
'is_monotonic_increasing',
'is_unique',
'isin',
'isna',
'isnull',
'item',
'items',
'iteritems',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lt',
'mad',
'map',
'mask',
'max',
'mean',
'median',
'memory_usage',
'min',
'mod',
'mode',
'mul',
'multiply',
'name',
'nbytes',
'ndim',
'ne',
'nlargest',
'notna',
'notnull',
```

```
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'radd',
'rank',
'ravel',
'rdiv',
'rdivmod',
'reindex',
'reindex_like',
'rename',
'rename_axis',
'reorder_levels',
'repeat',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'searchsorted',
'sem',
'set_axis',
'set_flags',
'shape',
'shift',
'size',
'skew',
'slice_shift',
'sort_index',
'sort_values',
'squeeze',
'std',
'sub',
'subtract',
'sum',
```

```
 'swapaxes',
 'swaplevel',
 'tail',
 'take',
 'to_clipboard',
 'to_csv',
 'to_dict',
 'to_excel',
 'to_frame',
 'to_hdf',
 'to_json',
 'to_latex',
 'to_list',
 'to_markdown',
 'to_numpy',
 'to_period',
 'to_pickle',
 'to_sql',
 'to_string',
 'to_timestamp',
 'to_xarray',
 'transform',
 'transpose',
 'truediv',
 'truncate',
 'tz_convert',
 'tz_localize',
 'unique',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'view',
 'where',
 'xs']
```

```
s=pd.Series(np.random.rand(5))
s
```

```
0    0.015646
1    0.977334
2    0.152281
3    0.914101
4    0.669473
dtype: float64
```

```
s[2:4]
```

```
2    0.152281
3    0.914101
dtype: float64
```

```
s.index = ['a','b','c','d','e']
```

```
s
```

```
a    0.015646
b    0.977334
c    0.152281
d    0.914101
e    0.669473
dtype: float64
```

```
s['e']
```

```
0.6694728328999546
```

 # DataFrame

```
s1=pd.Series(np.random.rand(5))
s2=pd.Series(np.random.rand(5))
s3=pd.Series(np.random.rand(5))
s4=pd.Series(np.random.rand(5))
```

```
s1
```

```
0    0.985379
1    0.670913
2    0.071290
3    0.029618
4    0.737482
dtype: float64
```

```
df = pd.DataFrame([s1,s2,s3,s4])
df
```

	0	1	2	3	4
0	0.985379	0.670913	0.071290	0.029618	0.737482
1	0.399835	0.018559	0.093672	0.713750	0.654045
2	0.454438	0.073028	0.164785	0.184355	0.768255
3	0.924023	0.570724	0.234560	0.960752	0.765374

```
df = df.T
df
```

	0	1	2	3
0	0.985379	0.399835	0.454438	0.924023
1	0.670913	0.018559	0.073028	0.570724
2	0.071290	0.093672	0.164785	0.234560
3	0.029618	0.713750	0.184355	0.960752
4	0.737482	0.654045	0.768255	0.765374

```
label = ['S1','S2','S3','S4']
label
```

```
['S1', 'S2', 'S3', 'S4']
```

```
df.columns=label
```

```
df
```

```
         S1        S2        S3        S4
0  0.985379  0.399835  0.454438  0.924023
1  0.670913  0.018559  0.073028  0.570724
2  0.071290  0.093672  0.164785  0.234560
3  0.029618  0.713750  0.184355  0.960752
4  0.737482 0.654045 0.768255 0.765374
```

```
df[['S1','S2']]
```

```
         S1        S2
0  0.985379  0.399835
1  0.670913  0.018559
2  0.071290  0.093672
3  0.029618  0.713750
4  0.737482  0.654045
```

```
df
```

```
         S1        S2        S3        S4
0  0.985379  0.399835  0.454438  0.924023
1  0.670913  0.018559  0.073028  0.570724
2  0.071290  0.093672  0.164785  0.234560
3  0.029618  0.713750  0.184355  0.960752
4  0.737482  0.654045  0.768255  0.765374
```

 # LOC & ILOC [:, :]

```
df.iloc[0:2,0:3]
```

```
         S1        S2        S3
0  0.985379 0.399835 0.454438
1  0.670913 0.018559 0.073028
```

```
df.loc[0:1,'S1':'S3']
```

```
         S1        S2        S3
0  0.985379  0.399835  0.454438
1  0.670913  0.018559  0.073028
```

```
import numpy as np
import pandas as pd
```

```
d = {'Name':['Hari','Srikanth','Navya','Mahi'],
     'Age':[29,37,23,41],
```

```
     'Gender':['M','M','F','M'],
     'Salary':[np.NaN,45000,30000,35000]}

d
```

```
{'Name': ['Hari', 'Srikanth', 'Navya', 'Mahi'],
 'Age': [29, 37, 23, 41],
 'Gender': ['M', 'M', 'F', 'M'],
 'Salary': [nan, 45000, 30000, 35000]}
```

```
df = pd.DataFrame(d)
df
```

```
       Name  Age Gender   Salary
0      Hari   29      M      NaN
1  Srikanth   37      M  45000.0
2     Navya   23      F  30000.0
3      Mahi   41      M  35000.0
```

```
df.isnull()
```

```
    Name    Age  Gender  Salary
0  False  False   False    True
1  False  False   False   False
2  False  False   False   False
3  False  False   False   False
```

```
df.isnull().sum()
```

```
Name      0
Age       0
Gender    0
Salary    1
dtype: int64
```

```
df['Salary'].mean()
```

```
36666.666666666664
```

```
df['Salary']=df['Salary'].fillna(df['Salary'].mean())
```

```
df
```

```
       Name  Age Gender        Salary
0      Hari   29      M  36666.666667
1  Srikanth   37      M  45000.000000
2     Navya   23      F  30000.000000
3      Mahi   41      M  35000.000000
```

```
df1 = df.copy()
```

```
df1
```

```
       Name  Age Gender        Salary
0      Hari   29      M  36666.666667
1  Srikanth   37      M  45000.000000
2     Navya   23      F  30000.000000
3      Mahi   41      M  35000.000000
```

```
dir(df)
```

```
['Age',
 'Gender',
 'Name',
 'Salary',
 'T',
 '_AXIS_LEN',
 '_AXIS_ORDERS',
 '_AXIS_REVERSED',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__divmod__',
 '__doc__',
 '__eq__',
 '__finalize__',
 '__floordiv__',
 '__format__',
 '__ge__',
 '__getattr__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__iand__',
 '__ifloordiv__',
```

```
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__invert__',
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__matmul__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
```

```
'__xor__',
'_accessors',
'_accum_func',
'_add_numeric_operations',
'_agg_by_level',
'_agg_examples_doc',
'_agg_summary_and_see_also_doc',
'_align_frame',
'_align_series',
'_arith_method',
'_as_manager',
'_attrs',
'_box_col_values',
'_can_fast_transpose',
'_check_inplace_and_allows_duplicate_labels',
'_check_inplace_setting',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_combine_frame',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_axes_from_arguments',
'_construct_result',
'_constructor',
'_constructor_sliced',
'_convert',
'_count_level',
'_data',
'_dir_additions',
'_dir_deletions',
'_dispatch_frame_op',
'_drop_axis',
'_drop_labels_or_levels',
'_ensure_valid_index',
'_find_valid_index',
'_flags',
'_from_arrays',
'_from_mgr',
'_get_agg_axis',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
```

```
'_get_bool_data',
'_get_cleaned_column_resolvers',
'_get_column_array',
'_get_index_resolvers',
'_get_item_cache',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_value',
'_getitem_bool_array',
'_getitem_multilevel',
'_gotitem',
'_hidden_attrs',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_info_repr',
'_init_mgr',
'_inplace_method',
'_internal_names',
'_internal_names_set',
'_is_copy',
'_is_homogeneous_type',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_iset_item',
'_iset_item_mgr',
'_iset_not_inplace',
'_item_cache',
'_iter_column_arrays',
'_ixs',
'_join_compat',
'_logical_func',
'_logical_method',
'_maybe_cache_changed',
'_maybe_update_cacher',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_needs_reindex_multi',
'_protect_consolidate',
'_reduce',
'_reindex_axes',
'_reindex_columns',
'_reindex_index',
'_reindex_multi',
'_reindex_with_indexers',
```

```
'_replace_columnwise',
'_repr_data_resource_',
'_repr_fits_horizontal_',
'_repr_fits_vertical_',
'_repr_html_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_sanitize_column',
'_series',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
'_set_item',
'_set_item_frame_value',
'_set_item_mgr',
'_set_value',
'_setitem_array',
'_setitem_frame',
'_setitem_slice',
'_slice',
'_stat_axis',
'_stat_axis_name',
'_stat_axis_number',
'_stat_function',
'_stat_function_ddof',
'_take_with_is_copy',
'_to_dict_of_blocks',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'append',
'apply',
'applymap',
'asfreq',
'asof',
'assign',
'astype',
```

```
'at',
'at_time',
'attrs',
'axes',
'backfill',
'between_time',
'bfill',
'bool',
'boxplot',
'clip',
'columns',
'combine',
'combine_first',
'compare',
'convert_dtypes',
'copy',
'corr',
'corrwith',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'eval',
'ewm',
'expanding',
'explode',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
'floordiv',
'from_dict',
```

```
'from_records',
'ge',
'get',
'groupby',
'gt',
'head',
'hist',
'iat',
'idxmax',
'idxmin',
'iloc',
'index',
'infer_objects',
'info',
'insert',
'interpolate',
'isin',
'isna',
'isnull',
'items',
'iteritems',
'iterrows',
'itertuples',
'join',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lookup',
'lt',
'mad',
'mask',
'max',
'mean',
'median',
'melt',
'memory_usage',
'merge',
'min',
'mod',
'mode',
'mul',
'multiply',
'ndim',
'ne',
'nlargest',
'notna',
```

```
'notnull',
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'pivot',
'pivot_table',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'query',
'radd',
'rank',
'rdiv',
'reindex',
'reindex_like',
'rename',
'rename_axis',
'reorder_levels',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'select_dtypes',
'sem',
'set_axis',
'set_flags',
'set_index',
'shape',
'shift',
'size',
'skew',
'slice_shift',
'sort_index',
'sort_values',
'squeeze',
'stack',
'std',
```

```
 'style',
 'sub',
 'subtract',
 'sum',
 'swapaxes',
 'swaplevel',
 'tail',
 'take',
 'to_clipboard',
 'to_csv',
 'to_dict',
 'to_excel',
 'to_feather',
 'to_gbq',
 'to_hdf',
 'to_html',
 'to_json',
 'to_latex',
 'to_markdown',
 'to_numpy',
 'to_parquet',
 'to_period',
 'to_pickle',
 'to_records',
 'to_sql',
 'to_stata',
 'to_string',
 'to_timestamp',
 'to_xarray',
 'to_xml',
 'transform',
 'transpose',
 'truediv',
 'truncate',
 'tz_convert',
 'tz_localize',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'where',
 'xs']
```

```
df2=pd.concat([df,df1])
df2
```

	Name	Age	Gender	Salary
0	Hari	29	M	36666.666667
1	Srikanth	37	M	45000.000000
2	Navya	23	F	30000.000000

```
3      Mahi   41      M  35000.000000
0      Hari   29      M  36666.666667
1  Srikanth   37      M  45000.000000
2     Navya   23      F  30000.000000
3      Mahi   41      M  35000.000000
```

```
df2.reset_index(drop=True)
```

```
       Name  Age Gender        Salary
0      Hari   29      M  36666.666667
1  Srikanth   37      M  45000.000000
2     Navya   23      F  30000.000000
3      Mahi   41      M  35000.000000
4      Hari   29      M  36666.666667
5  Srikanth   37      M  45000.000000
6     Navya   23      F  30000.000000
7      Mahi   41      M  35000.000000
```

```
df2.drop_duplicates()
```

```
       Name  Age Gender        Salary
0      Hari   29      M  36666.666667
1  Srikanth   37      M  45000.000000
2     Navya   23      F  30000.000000
3      Mahi   41      M  35000.000000
```

```
pd.DataFrame()
```

```
df['Salary'].astype('int')
```

```
0    36666
1    45000
2    30000
3    35000
Name: Salary, dtype: int64
```

Data Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns
```

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

```
x
```

```
array([1, 2, 3, 4, 5, 6])
```

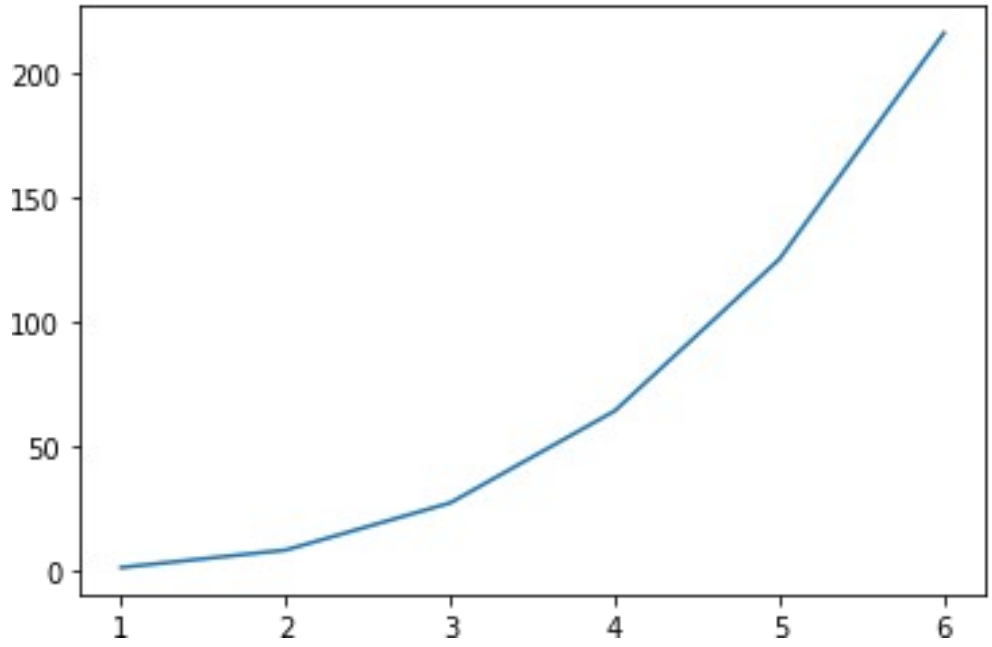
```
y = np.power(x,3)
```

```
y
```

```
array([  1,   8,  27,  64, 125, 216])
```

```
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fe44b403450>]
```




```
plt.style.available
```

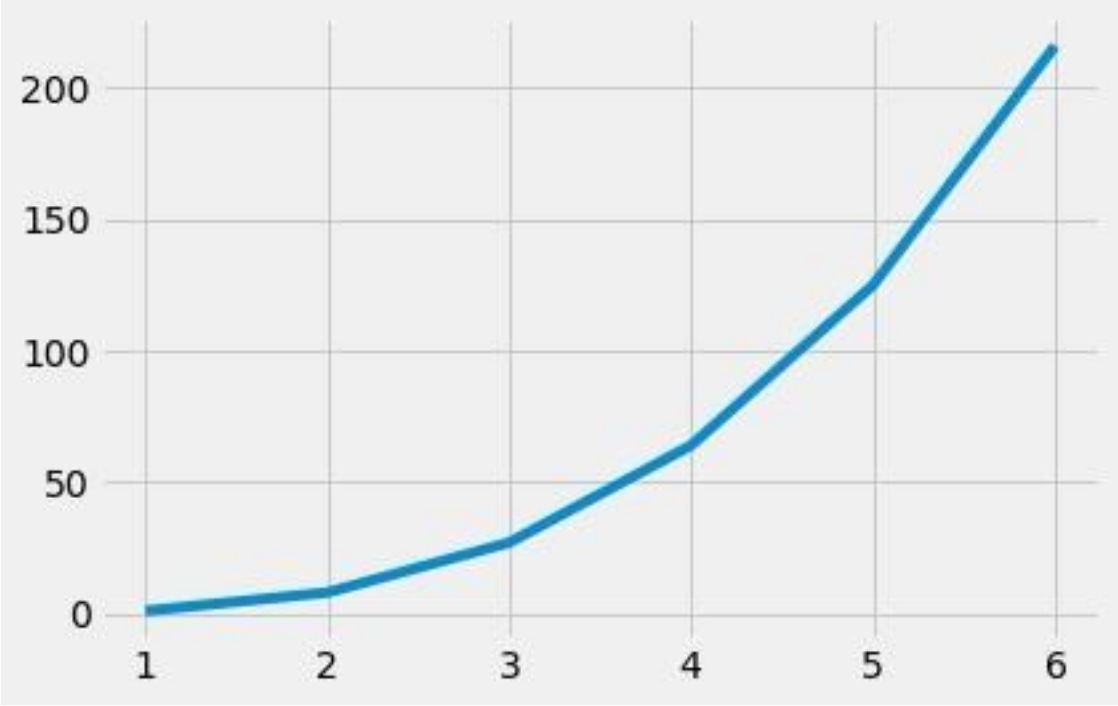
```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
```

```
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

```
plt.style.use('fivethirtyeight')
```

```
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fe44ae48e50>]
```




```
x = np.linspace(0,10,1000)
y = np.sin(x)
```

```
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fe44ae3bc50>]
```

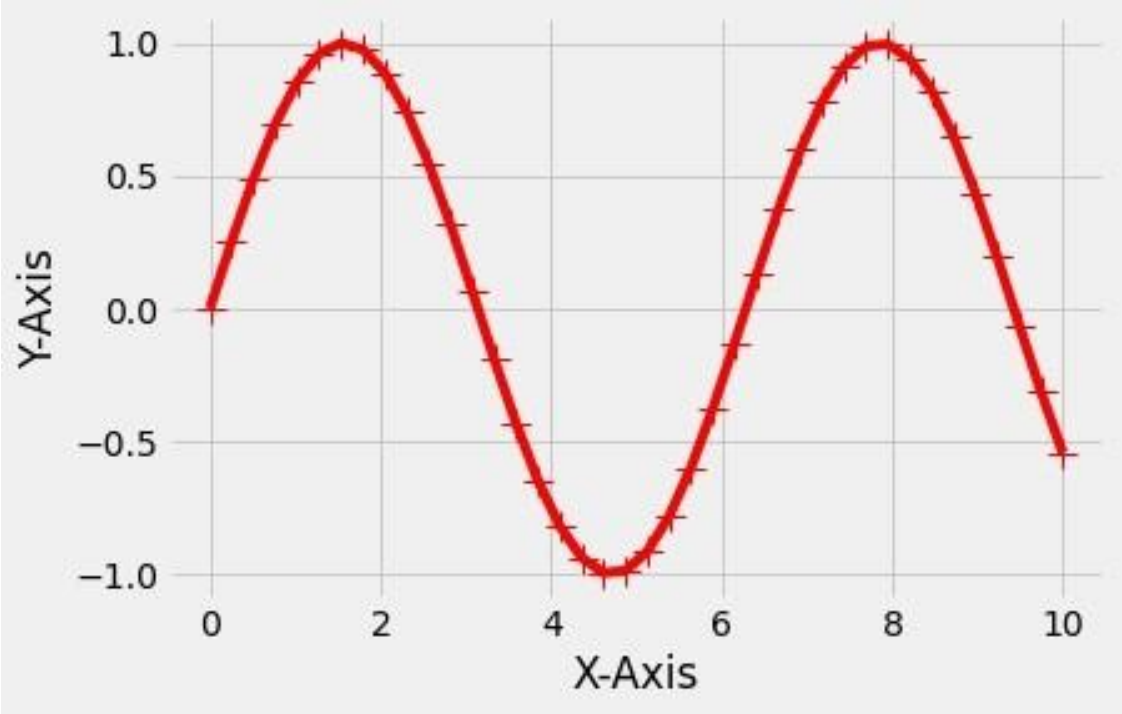

```
plt.plot(x,y,color='r',marker='o')
```

```
[<matplotlib.lines.Line2D at 0x7fe44acc4490>]
```


```
x = np.linspace(0,10,40)
y = np.sin(x)
```

```
plt.plot(x,y,color='r',marker='+',markersize=12)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
```

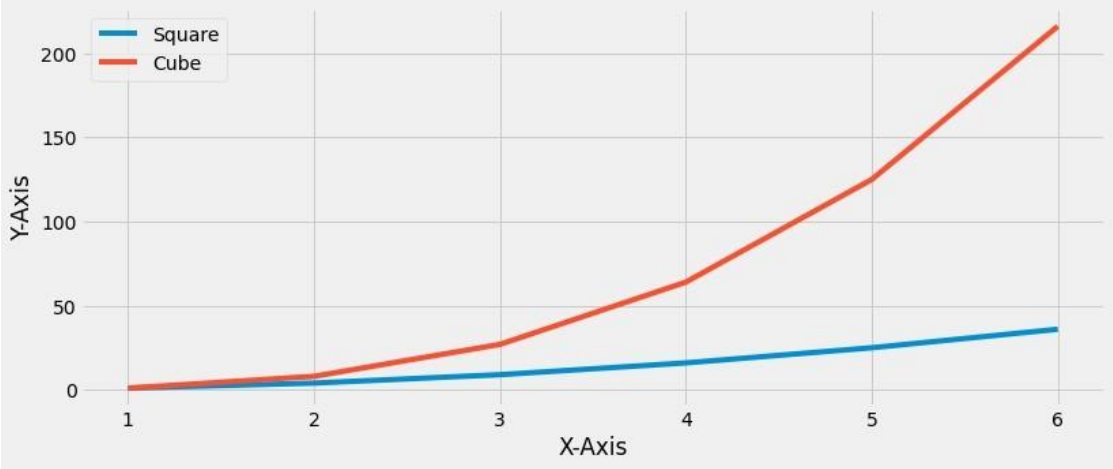
```
Text(0, 0.5, 'Y-Axis')
```




```
x = np.array([1,2,3,4,5,6])
y1 = np.power(x,2)
y2 = np.power(x,3)
```

```
plt.figure(figsize=(12,5))
plt.plot(x,y1,label='Square')
plt.plot(x,y2,label='Cube')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.legend()
```

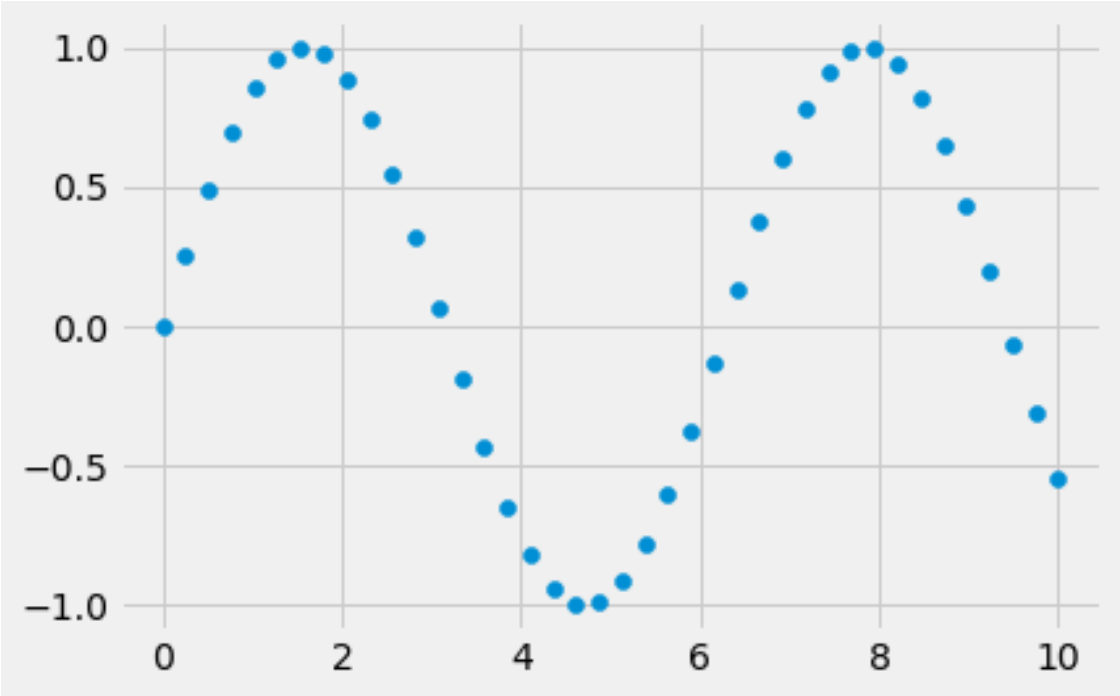
```
<matplotlib.legend.Legend at 0x7fe44a8e7a10>
```




```
x = np.linspace(0,10,40)
y = np.sin(x)

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7fe44a854e90>
```

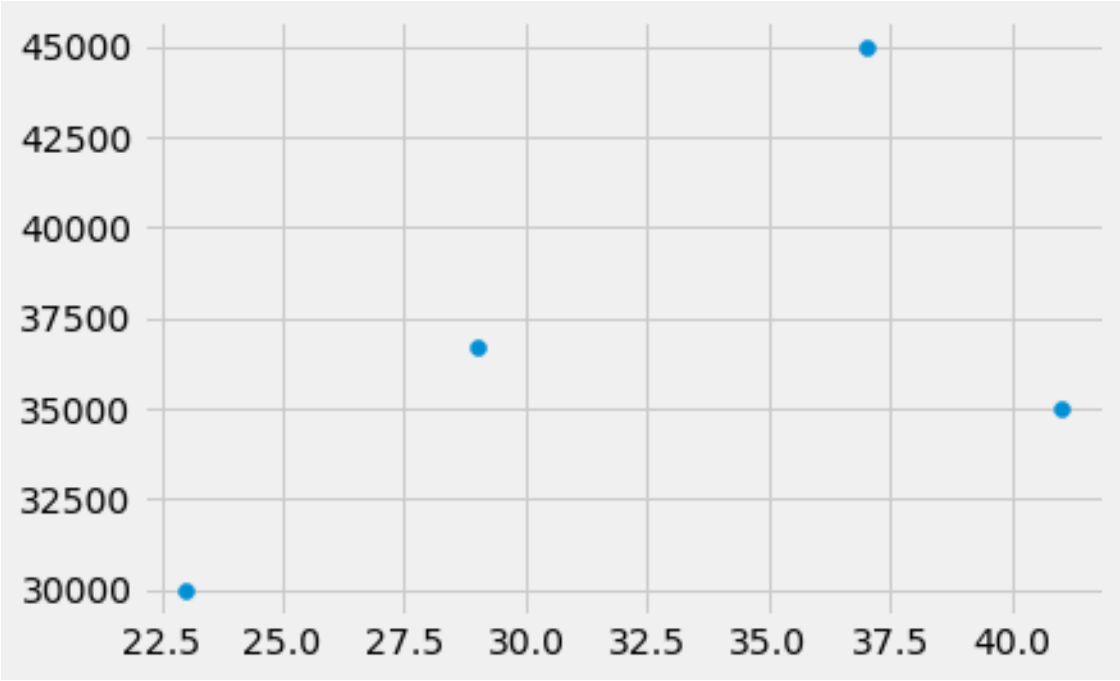



```
df
```

	Name	Age	Gender	Salary
0	Hari	29	M	36666.666667
1	Srikanth	37	M	45000.000000
2	Navya	23	F	30000.000000
3	Mahi	41	M	35000.000000

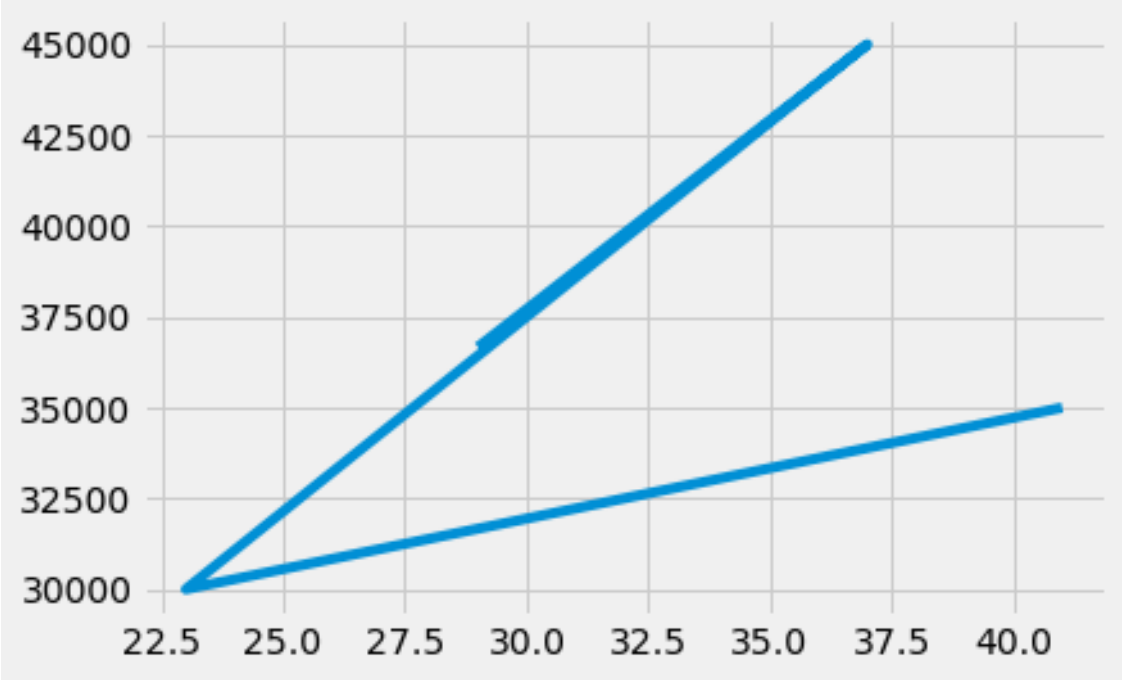
```
plt.scatter(df['Age'],df['Salary'])
```

```
<matplotlib.collections.PathCollection at 0x7fe44a815b50>
```




```
plt.plot(df['Age'],df['Salary'])
```

```
[<matplotlib.lines.Line2D at 0x7fe44abad750>]
```



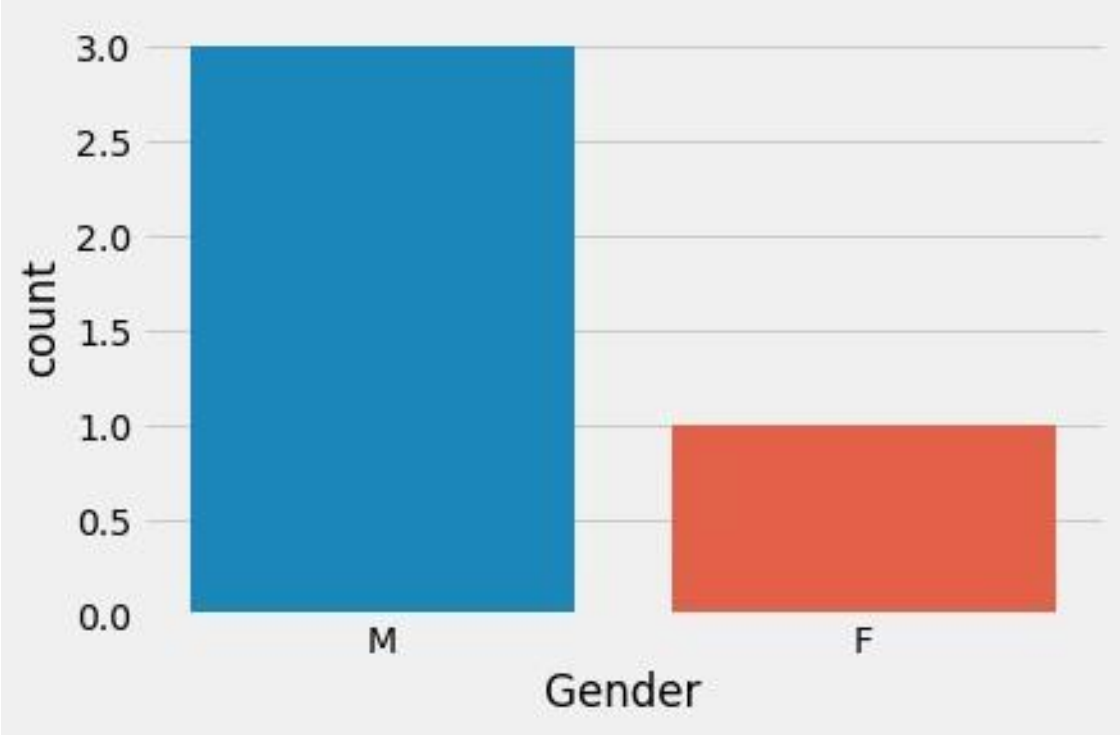

```
plt.bar(df['Gender'],df['Age'])
```

```
<BarContainer object of 4 artists>
```


```
sns.countplot(df['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

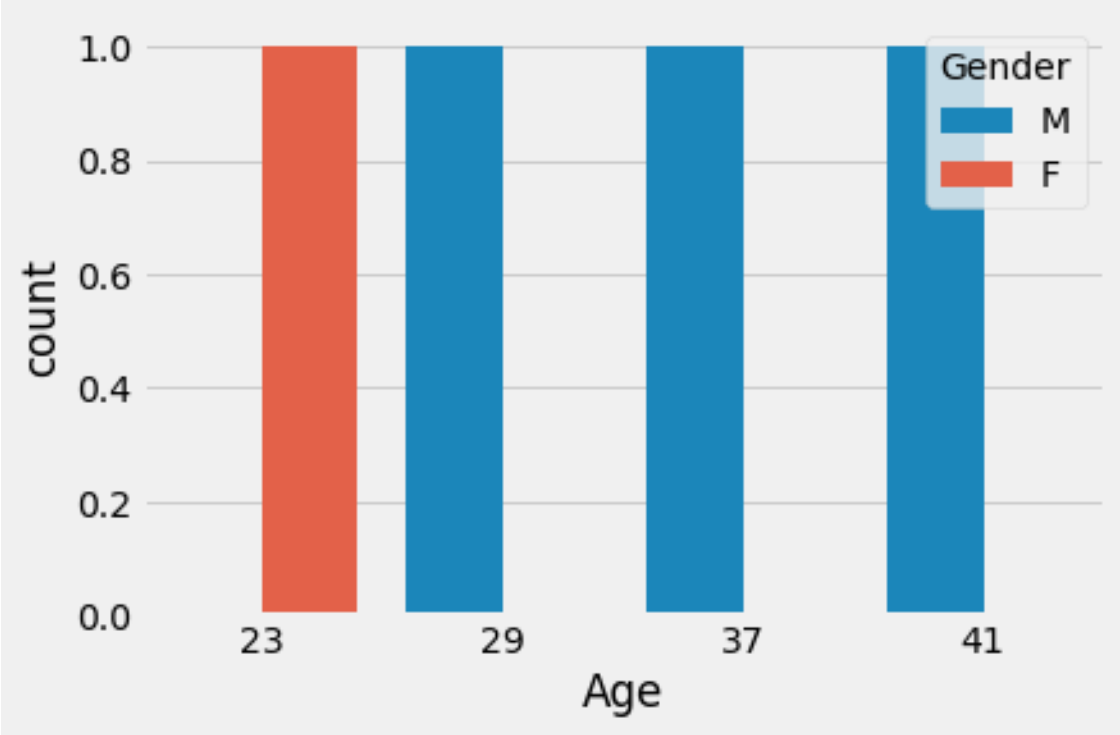
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe44a50f450>
```




```
sns.countplot(df['Age'],hue=df['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

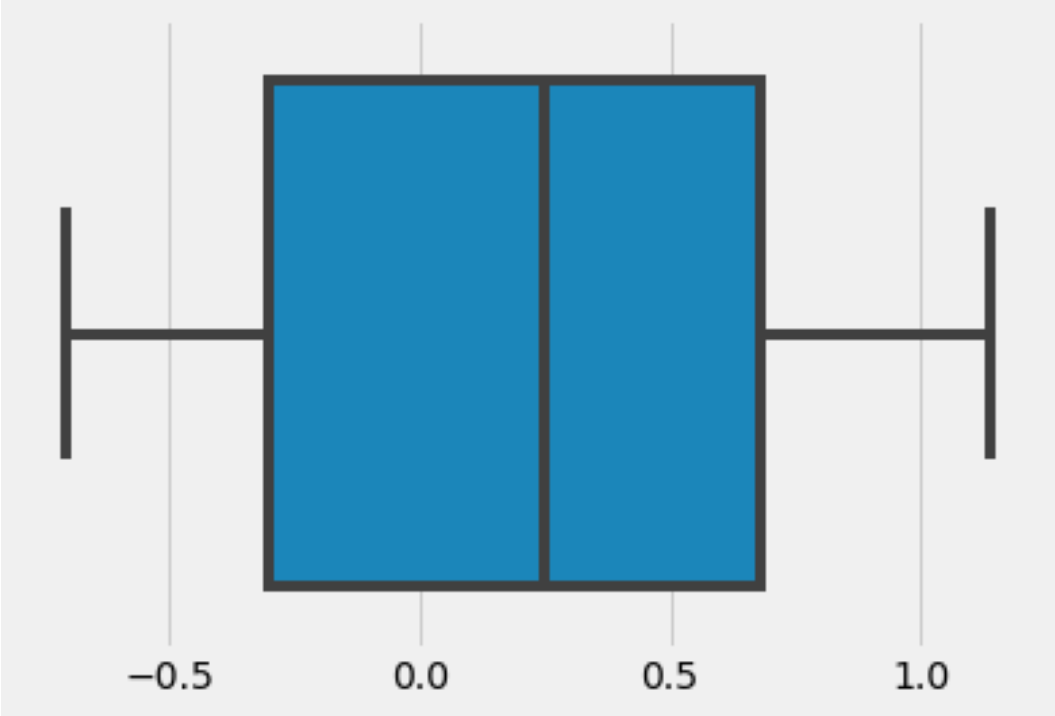
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe44a4dd250>
```




```
sns.boxplot(np.random.randn(6))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe44a45df10>
```



Data Wrangling & Data Pre-Processing

```
df = pd.read_csv('/content/Data1.csv')
```

```
df.head(6)
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
```

```
df.tail(3)
```

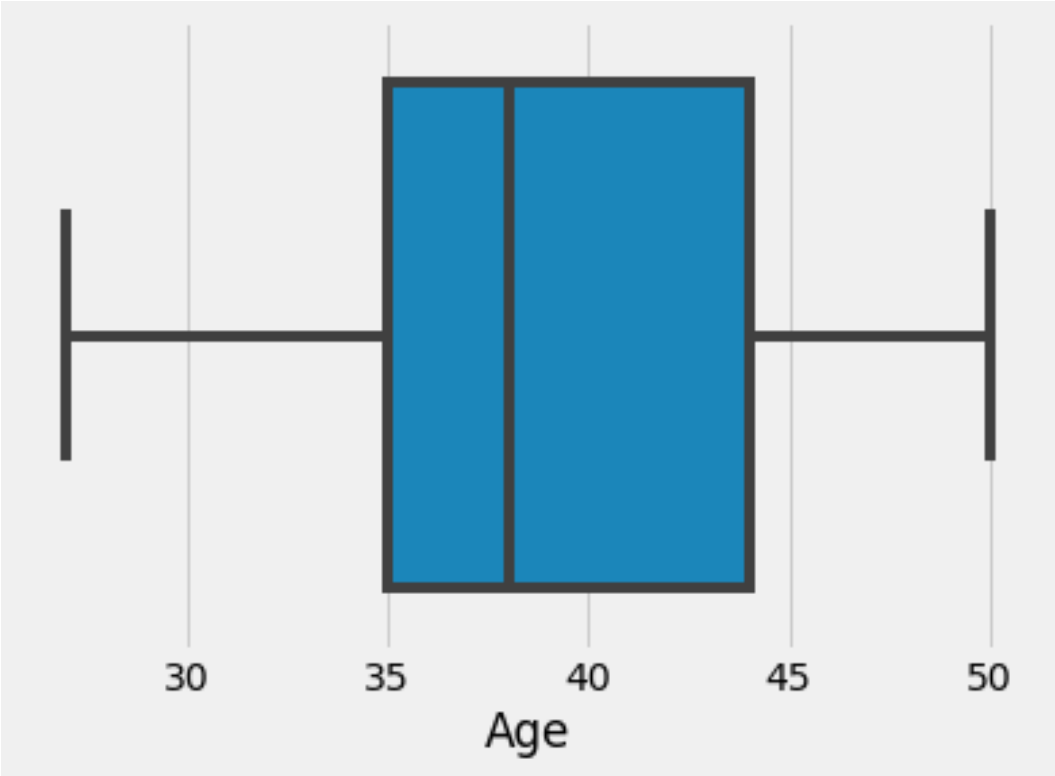
```
   Country   Age   Salary Purchased
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
```

```
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
```

```
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe44a45de90>
```




```
df.describe()
```

```
             Age        Salary
count   9.000000      9.000000
mean   38.777778  63777.777778
std     7.693793  12265.579662
min    27.000000  48000.000000
25%    35.000000  54000.000000
50%    38.000000  61000.000000
75%    44.000000  72000.000000
max    50.000000  83000.000000
```

```
df.describe(include='all')
```

```
       Country        Age        Salary Purchased
count       10   9.000000      9.000000        10
unique       3        NaN           NaN         2
top     France        NaN           NaN        No
freq         4        NaN           NaN         5
mean       NaN  38.777778  63777.777778       NaN
std        NaN   7.693793  12265.579662       NaN
min        NaN  27.000000  48000.000000       NaN
25%        NaN  35.000000  54000.000000       NaN
```

```
50%         NaN  38.000000  61000.000000       NaN
75%         NaN  44.000000  72000.000000       NaN
max         NaN  50.000000  83000.000000       NaN
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

```
df.isnull().sum()
```

```
Country      0
Age          1
Salary       1
Purchased    0
dtype: int64
```

```
df.head(7)
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
```

```
df['Salary'].mean()
```

```
63777.77777777778
```

```
df['Age'].median()
```

```
38.0
```

```
df['Salary']=df['Salary'].fillna(df['Salary'].mean())
```

```
df['Age']=df['Age'].fillna(df['Age'].median())
```

```
df
```

```
   Country   Age        Salary Purchased
0   France  44.0  72000.000000        No
```

```
1    Spain  27.0  48000.000000       Yes
2  Germany  30.0  54000.000000        No
3    Spain  38.0  61000.000000        No
4  Germany  40.0  63777.777778       Yes
5   France  35.0  58000.000000       Yes
6    Spain  38.0  52000.000000        No
7   France  48.0  79000.000000       Yes
8  Germany  50.0  83000.000000        No
9   France  37.0  67000.000000       Yes
```

```
df['Age']>40
```

```
0     True
1    False
2    False
3    False
4    False
5    False
6    False
7     True
8     True
9    False
Name: Age, dtype: bool
```

```
df[df['Age']>40]
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No

```
df.loc[df['Age']>40]
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No

```
df[df['Salary']>70000]
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No

Encoding

 # Method 1 (One Hot Encoding)

```
country = pd.get_dummies(df['Country'])
country
```

```
   France  Germany  Spain
0       1        0      0
1       0        0      1
2       0        1      0
3       0        0      1
4       0        1      0
5       1        0      0
6       0        0      1
7       1        0      0
8       0        1      0
9       1        0      0
```

```
df.join(country)
```

```
   Country   Age        Salary Purchased  France  Germany  Spain
0   France  44.0  72000.000000        No       1        0      0
1    Spain  27.0  48000.000000       Yes       0        0      1
2  Germany  30.0  54000.000000        No       0        1      0
3    Spain  38.0  61000.000000        No       0        0      1
4  Germany  40.0  63777.777778       Yes       0        1      0
5   France  35.0  58000.000000       Yes       1        0      0
6    Spain  38.0  52000.000000        No       0        0      1
7   France  48.0  79000.000000       Yes       1        0      0
8  Germany  50.0  83000.000000        No       0        1      0
9   France  37.0  67000.000000       Yes       1        0      0
```

 # pd.concat([df,country])

 # Method 2 (Label Encoding)

```
from sklearn.preprocessing import LabelEncoder
from collections import Counter as count
```

```
count(df['Country'])
```

```
Counter({'France': 4, 'Spain': 3, 'Germany': 3})
```

```
le = LabelEncoder()
```

```
df['Country'] = le.fit_transform(df['Country'])
```

```
count(df['Country'])
```

```
Counter({0: 4, 2: 3, 1: 3})
```

```
df
```

```
   Country   Age        Salary Purchased
0        0  44.0  72000.000000        No
1        2  27.0  48000.000000       Yes
2        1  30.0  54000.000000        No
3        2  38.0  61000.000000        No
4        1  40.0  63777.777778       Yes
```

```
5         0  35.0  58000.000000       Yes
6         2  38.0  52000.000000        No
7         0  48.0  79000.000000       Yes
8         1  50.0  83000.000000        No
9         0  37.0  67000.000000       Yes
```

 # method 3 (feature map) df['Purchased']=df['Purchased'].replace(['No','Yes'],[0,1])df

```
   Country   Age        Salary  Purchased
0        0  44.0  72000.000000          0
1        2  27.0  48000.000000          1
2        1  30.0  54000.000000          0
3        2  38.0  61000.000000          0
4        1  40.0  63777.777778          1
5        0  35.0  58000.000000          1
6        2  38.0  52000.000000          0
7        0  48.0  79000.000000          1
8        1  50.0  83000.000000          0
9        0  37.0  67000.000000          1
```

Spliting the data

```
x = df.iloc[:,0:3]
x
```

```
   Country   Age        Salary
0        0  44.0  72000.000000
1        2  27.0  48000.000000
2        1  30.0  54000.000000
3        2  38.0  61000.000000
4        1  40.0  63777.777778
5        0  35.0  58000.000000
6        2  38.0  52000.000000
7        0  48.0  79000.000000
8        1  50.0  83000.000000
9        0  37.0  67000.000000
```

```
y = df['Purchased']
y
```

```
0    0
1    1
2    0
3    0
4    1
5    1
6    0
7    1
8    0
```

```
9    1
Name: Purchased, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest =
train_test_split(x,y,test_size=0.3,random_state=11)
```

```
xtrain
```

```
   Country   Age        Salary
6        2  38.0  52000.000000
4        1  40.0  63777.777778
5        0  35.0  58000.000000
1        2  27.0  48000.000000
3        2  38.0  61000.000000
0        0  44.0  72000.000000
9        0  37.0  67000.000000
```

```
xtest
```

```
   Country   Age   Salary
7        0  48.0  79000.0
8        1  50.0  83000.0
2        1  30.0  54000.0
```

Scaling

 # Normalization ---> output(0 to 1)

```
from sklearn.preprocessing import MinMaxScaler
```

```
nm = MinMaxScaler()
```

```
n_xtrain = nm.fit_transform(xtrain)
n_xtrain
```

```
array([[1.        , 0.64705882, 0.16666667],
       [0.5       , 0.76470588, 0.65740741],
       [0.        , 0.47058824, 0.41666667],
       [1.        , 0.        , 0.        ],
       [1.        , 0.64705882, 0.54166667],
       [0.        , 1.        , 1.        ],
       [0.        , 0.58823529, 0.79166667]])
```

```
n_xtest = nm.transform(xtest)
n_xtest
```

```
array([[0.        , 1.23529412, 1.29166667],
       [0.5       , 1.35294118, 1.45833333],
       [0.5       , 0.17647059, 0.25      ]])
```