In [ ]:
```
# id(), type(), len(),
# append(), index(), pop(), remove(), insert() clear(), range(), copy(), extend()
# map(), filter(), zip(), next(), reduce(), pow(),
# iterator(), generator(),decorator(),
# lamda(), all(), round(), sort(), def(), import(), math(), open()
# upper(), lower(), max(), min(),sum(),
# capitalize()
# config() or configure(), get(), randint()
```

In [ ]:
```
# '''
# import statistics
# from math import sqrt, pi, log10....Provides mathematical functions and constants
# import datetime... For working with dates and times.
# import json...For working with JSON (JavaScript Object Notation) data
# import re... For regular expressions.
# import collections... Provides specialized container datatypes (e.g., defaultdict
# import tkinter as tk
# import pandas as pd...For data manipulation and analysis.
# import numpy as np ...For numerical computing.
# import matplotlib.pyplot...For plotting and visualization.
# import requests...For making HTTP requests.
# import random...For generating random numbers.
# import os .....For interacting with the operating system (e.g., file paths, direc
# import sys....Provides access to system-specific parameters and functions (e.g.,
# import mysql.connector
# from mysql.connector import error
# import pymsql
# import flask as Flask
# from my_module import greet'''
```

In [1]:
```
print('test')
```

```
test
```

In [2]:
```
a = "hi"
print(a)
```

```
hi
```

In [104…
```
a=[1,5,2,9,8]
print(sum(a))
print(f'{a.append(6)}')
```

```
25
None
```

In [63]:
```
b=[7,8,9,2,1]  # Iterator
c=b.__iter__()
print(c.__next__())
print(c.__next__())
print(c.__next__())
print(c.__next__())
```

```
7
8
9
2
```

In [19]:
```
b=[7,8,9,2,1] # Iterator
c=iter(b)
print(next(c))
print(next(c))
```

```
7
8
```

In [51]:
```
def my_generator(): # Generator
        yield 1
        yield 2
        yield 3
        yield 4

gen = my_generator()
print(next(gen))
print(next(gen))
print(next(gen))
print(next(gen))
```

```
1
2
3
4
```

In [62]:
```
def num():    # Generator
    for i in range(1,5):
        yield i

a = num()
print(a.__next__())
print(a.__next__())
print(a.__next__())
```

```
1
2
3
```

In [52]:
```
def my_generator(): # Generator
        yield 1
        yield 2
        yield 3
        yield 4

gen = my_generator()
print(gen.__next__())
print(gen.__next__())
print(gen.__next__())
print(gen.__next__())
```

```
1
2
3
4
```

In [59]:
```
def num():
    for i in range(1,5):
        return i

print(num())
```

```
1
```

In [18]:
```
a1=[1,2,4,5,8]
a2=['a','b','c','d','e','f']
a3=list(zip(a1,a2))
print(a3)
```

```
[(1, 'a'), (2, 'b'), (4, 'c'), (5, 'd'), (8, 'e')]
```

In [38]:
```
def mul(a,b):
    c=a*b
    print(c)

mul(4, 6)
```

```
24
```

In [39]:
```
a = [1, 2, 3, 4, 5, 6] # lambda function
even_numbers = list(filter(lambda x: x % 2 == 0, a))
print(even_numbers)
```

```
[2, 4, 6]
```

In [23]:
```
def multiplier(n): # lambda function
    return lambda a: a * n

doubler = multiplier(2)
tripler = multiplier(3)
print(doubler(10))
print(tripler(10))
```

```
20
30
```

In [2]:
```
def la(x):
    return (lambda a: a**2)(x)

print(la(4))
```

```
16
```

In [34]:
```
add = lambda a,b: a+b # lambda function
print(add(5,4))
```

```
9
```

In [37]:
```
def mul(a,b):
    return a*b
```

```
print(mul(4, 6))
```

```
24
```

In [92]:
```
def dec_mul(func): #Decorator
    def wrapper():
        print('*'*20)
        func()
        print('_'*20)
        return
    return wrapper

@dec_mul
def mul():
    a=4
    b=6
    c = a*b
    print(c)

mul()
```

```
********************
24
____________________
```

In [48]:
```
print('/'*20)
```

```
////////////////////
```

In [6]:
```
class Person:                           #..........Defining a Class:
    def __init__(self, name, age):     #.......The __init__ Method (Constructor):
        self.name = name ## Initialize the 'name' attribute
        self.age = age

    def myfunc(self):                    #.........Instance Methods:
        print("Hello my name is " + self.name)

p1 = Person("John", 36)                 #....Creating Objects (Instantiating the Cl

#..........Accessing Attributes and Calling Methods:
print(p1.name)   # prints the name
print(p1.age)    # prints the age
p1.myfunc()      # calls the method
```

```
John
36
Hello my name is John
```

In [67]:
```
def mul(a,b):
    c = a*b
    print(c)

mul(4,6)
```

```
24
```

In [79]:
```
def ad(a,b): # map function
    return a+b

x=map(ad, ('hello','hi'),('HELLO','HI'))

print(list(x))
```

```
['helloHELLO', 'hiHI']
```

In [80]:
```
def ad(a,b): # map function
    return a+b

x=map(ad, ('hello'),('HELLO'))

print(list(x))
```

```
['hH', 'eE', 'lL', 'lL', 'oO']
```

In [47]:
```
a = [1, 2, 3] # map function
b = [4, 5, 6]
res = map(lambda x, y: x + y, a, b)
print(list(res))
```

```
[5, 7, 9]
```

In [4]:
```
def ad(a,b): # map function
    return (lambda d, c: d+c)(a,b)

x=map(ad, ('hello'),('HELLO'))

print(list(x))
```

```
['hH', 'eE', 'lL', 'lL', 'oO']
```

In [86]:
```
    numbers = [1, 2, 3, 4, 5]
    squares_dict = {num: num**2 for num in numbers}
    print(squares_dict)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [88]:
```
squares = [] # .....Comprehensions
even_numbers = []
squares = [x**2 for x in range(10)]
even_numbers = [x for x in range(20) if x % 2 == 0]
print(list(squares))
print(list(even_numbers))
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [91]:
```
name_age_dict = [] # .....Comprehensions
names = ["Alice", "Bob", "Charlie"]
ages = [30, 24, 35]
name_age_dict = {name:age for name, age in zip(names, ages)}
print(dict(name_age_dict))
```

```
{'Alice': 30, 'Bob': 24, 'Charlie': 35}
```

In [94]:
```
unique_letters = {char for char in "hello world" if char.isalpha()}
print(unique_letters)
```

```
{'h', 'r', 'w', 'e', 'l', 'd', 'o'}
```

In [2]:
```
a = '''hello world
       I am Vishnu
       Studying in 1st std'''
# lines = a.readlines()
line_count = len(a)
print(f"Total lines: {line_count}")
```

```
Total lines: 59
```

In [ ]:
```
.......Flask
```

In [ ]:
```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "Welcome to E-Commerce'd!"

<style></style>
<h></h>
<h1></h1>
<h2></h2>
<body></body>
<div></div>
<p></p>
<ul></ul>
<ol></ol>
<form></form>
<br>
<a href="/Signout" style="text-decoration: none;">
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
<button style="background-color: darkblue; color: white; border: none; padding: 8px
<img src="static/Rose.jpeg" alt="AI Domains" style="width: 100px; margin-bottom: 10

if __name__ == "__main__":
    app.run(debug=True)
```

In [3]:
```
import random

# Generate a random integer between 1 and 10 (inclusive)
random_number_1 = random.randint(1, 10)
print(f"Random number between 1 and 10: {random_number_1}")

# Generate a random integer between 50 and 100 (inclusive)
random_number_2 = random.randint(50, 100)
print(f"Random number between 50 and 100: {random_number_2}")

# Generate a random integer for a dice roll (1 to 6 inclusive)
```

```
dice_roll = random.randint(1, 6)
print(f"Dice roll: {dice_roll}")
```

```
Random number between 1 and 10: 5
Random number between 50 and 100: 99
Dice roll: 4
```

In [ ]:
```
..............numpy___________
```

In [19]:
```
import numpy as np
a = np.array([1,2,3,4,5])
print(a.shape) # shape of the array: (5,)
print(a.dtype) # type of the elements: int32
print(a.ndim) # number of dimensions: 1
print(a.size) # total number of elements: 5
print(a.itemsize) # the size in bytes of each element: 4
```

```
(5,)
int64
1
5
8
```

In [21]:
```
a = np.array([1,2,3]) #......np array
b = a * np.array([2,0,2])
c = a + np.array([2,0,2])
a2 = a + np.array([4]) # this is called broadcasting, adds 4 to each element
a3 = a + np.array([4,4,4])
a4 = 2 * a # multiplication for each element
a5 = np.sqrt(a) # np.exp(a), np.tanh(a)
a6 = np.log(a)
print(b)
print(c)
print(a2)
print(a3)
print(a4)
print(a5) # [1. 1.41421356 1.73205081]
print(a6)
l = [1,2,3] #......List
l.append(4)
l2 = l + [5]
print(l2)
l3 = l*2 # list l repeated 2 times, same a l+l
print(l3)
l4 = [] # modify each item in the list
for i in l:
    l4.append(i**2)
print(l4)
l5 = [i**2 for i in l] # list comprehension
print(l5)
```

```
[2 0 6]
[3 2 5]
[5 6 7]
[5 6 7]
[2 4 6]
[1.         1.41421356 1.73205081]
[0.         0.69314718 1.09861229]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 1, 2, 3, 4]
[1, 4, 9, 16]
[1, 4, 9, 16]
```

In [35]:
```python
import numpy as np
a = np.array([1,2])
b = np.array([3,4])

dot = 0                 # numbersome way for lists
for i in range(len(a)):
    dot += a[i] * b[i]
print(dot) # 11

dot = np.dot(a,b)          # easy with numpy :)
print(dot) # 11

c = a * b                  # step by step manually
print(c) # [3 8]
d = np.sum(c)
print(d) # 11

dot = a.dot(b)               # most of these functions are also instance methods
print(dot) # 11
dot = (a*b).sum()
print(dot) # 11

dot = a @ b                  # in newer versions
print(dot) # 11
```

```
11
11
[3 8]
11
11
11
11
```

In [6]:
```python
import numpy as np
from timeit import default_timer as timer

a = np.random.randn(1000)
b = np.random.randn(1000)
A = list(a)
B = list(b)
T = 1000
def dot1():
    dot = 0
    for i in range(len(A)):
```

```
        dot += A[i]*B[i]
    return dot
def dot2():
    return np.dot(a,b)
start = timer()
for t in range(T):
    dot1()
end = timer()
t1 = end-start
start = timer()
for t in range(T):
    dot2()
end = timer()
t2 = end-start
print('Time with lists:', t1) # -> 0.19371
print('Time with array:', t2) # -> 0.00112
print('Ratio', t1/t2) # -> 172.332 times faster
```

```
Time with lists: 0.45732479999969655
Time with array: 0.006903599999532162
Ratio 66.2443942335431
```

In [16]:

```
a = np.array([[1,2], [3,4]]) # (matrix class exists but not recommended to use)
print(a)
print(a.shape) # (2, 2)

print(a[0]) # [1 2]             # Access elements
print(a[0][0]) # 1              # row first, then columns
print(a[0,0]) # 1               # row first, then columns
print(a[:,0])                   # slicing # all rows in col 0: [1 3]
print(a[0,:])                   # all columns in row 0: [1 2]

a.T                              # transpose
b = np.array([[3, 4], [5,6]])     # matrix multiplication
c = a.dot(b)
d = a * b                         # elementwise multiplication
b = np.array([[2,3], [4,6]])   # inner dimensions must match!
c = a.dot(b.T)
c = np.linalg.det(a)               # determinant
c = np.linalg.inv(a)              # inverse
c = np.diag(a)                    # diag
print(c)                          # [1 4]

c = np.diag([1,4])                # diag on a vector returns diagonal matrix (over
print(c)

# print(a[bool_idx]) # [3 4 5 6]     # note: this will be a rank 1 array!

print(a[a > 2]) # [3 4 5 6]      # We can do all of the above in a single concise s

b = np.where(a>2, a, -1)          # np.where(): same size with modified values
print(b)

a = np.array([10,19,30,41,50,61])   # fancy indexing: access multiple indices at on
b = a[[1,3,5]]
print(b) # [19 41 61]
```

```
even = np.argwhere(a%2==0).flatten()  # compute indices where condition is True
print(even) # [0 2 4]
a_even = a[even]
print(a_even) # [10 30 50]
```

```
[[1 2]
 [3 4]]
(2, 2)
[1 2]
1
1
[1 3]
[1 2]
[1 4]
[[1 0]
 [0 4]]
[3 4]
[[-1 -1]
 [ 3  4]]
[19 41 61]
[0 2 4]
[10 30 50]
```

In [22]:
```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

b = a[0,1]                              # Integer array indexing
print(b)

row0 = a[0,:]                            # Slicing
print(row0)                              # [1 2 3 4]
col0 = a[:, 0]
print(col0)                               # [1 5 9]
slice_a = a[0:2,1:3]
print(slice_a)

last = a[-1,-1]                    # indexing starting from the end: -1, -2 etc...
print(last) # 12

a = np.array([[1,2], [3, 4], [5, 6]])    # Boolean indexing:
print(a)

bool_idx = a > 2                   # same shape with True or False for the condition
print(bool_idx)
```

```
2
[1 2 3 4]
[1 5 9]
[[2 3]
 [6 7]]
12
[[1 2]
 [3 4]
 [5 6]]
[[False False]
 [ True  True]
 [ True  True]]
```

In [11]:
```python
a = np.arange(1, 7)
print(a)
b = a.reshape((2, 3)) # error if shape cannot be used
print(b)

c = a.reshape((3, 2)) # 3 rows, 2 columns
print(c)

print(a.shape) # (6,)
d = a[np.newaxis, :]
print(d) # [[1 2 3 4 5 6]]
print(d.shape) # (1, 6)
e = a[:, np.newaxis]
print(e)
print(e.shape) # (6, 1)
```

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

In [23]:
```python
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6]])

c = np.concatenate((a, b), axis=None)   # combine into 1d
print(c) # [1 2 3 4 5 6]

d = np.concatenate((a, b), axis=0)       # add new row
print(d)

e = np.concatenate((a, b.T), axis=1)  # add new column: note that we have to transp
```

```
print(e)

a = np.array([1,2,3,4])  # hstack: Stack arrays in sequence horizontally (column wi
b = np.array([5,6,7,8])
c = np.hstack((a,b))
print(c) # [1 2 3 4 5 6 7 8]

a = np.array([[1,2], [3,4]]) # hstack: Stack arrays adding one more row horizontall
b = np.array([[5,6], [7,8]])
c = np.hstack((a,b))
print(c)

a = np.array([1,2,3,4])  # vstack: Stack arrays in sequence vertically (row wise).
b = np.array([5,6,7,8])
c = np.vstack((a,b))
print(c)

a = np.array([[1,2], [3,4]])
b = np.array([[5,6], [7,8]])
c = np.vstack((a,b))
print(c)

x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
y = np.array([1, 0, 1])
z = x + y # Add v to each row of x using broadcasting
print(z)
```

```
[1 2 3 4 5 6]
[[1 2]
 [3 4]
 [5 6]]
[[1 2 5]
 [3 4 6]]
[1 2 3 4 5 6 7 8]
[[1 2 5 6]
 [3 4 7 8]]
[[1 2 3 4]
 [5 6 7 8]]
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]
```

In [ ]:
```
a = np.array([[7,8,9,10,11,12,13], [17,18,19,20,21,22,23]])
print(a.sum()) # default=None-> 210
print(a.sum(axis=None)) # overall sum -> 210
print(a.sum(axis=0)) # along the rows -> 1 sum entry for each column
# -> [24 26 28 30 32 34 36]
print(a.sum(axis=1)) # along the columns -> 1 sum entry for each row # -> [ 70 140]

print(a.mean()) # default=None-> 15.0
print(a.mean(axis=None)) # overall mean -> 15.0
```

```
print(a.mean(axis=0)) # along the rows -> 1 mean entry for each column
# -> [12. 13. 14. 15. 16. 17. 18.]
print(a.mean(axis=1)) # along the columns -> 1 mean entry for each row
# -> [10. 20.]
# some more: std, var, min, max
```

In [ ]:
```
x = np.array([1, 2])    # Let numpy choose the datatype
print(x.dtype) # int32

x = np.array([1.0, 2.0])  # Let numpy choose the datatype
print(x.dtype) # float64

x = np.array([1, 2], dtype=np.int64) # 8 bytes # Force a particular datatype, how m
print(x.dtype) # int64
x = np.array([1, 2], dtype=np.float32) # 4 bytes
print(x.dtype) # float32
```

In [25]:
```
a = np.array([1,2,3])
b = a # only copies reference!
b[0] = 42
print(a) # [42 2 3]
a = np.array([1,2,3])
b = a.copy() # actual copy!
b[0] = 42
print(a) # [1 2 3]
```

```
[42  2  3]
[1 2 3]
```

In [ ]:
```
a = np.zeros((2,3)) # size as tuple   # zeros

b = np.ones((2,3))            # ones

c = np.full((3,3),5.0)        # specific value

d = np.eye(3) #3x3           # identity

e = np.arange(10)             # arange

f = np.linspace(0, 10, 5)     # linspace
```

In [ ]:
```
a = np.random.random((3,2)) # uniform 0-1 distribution

b = np.random.randn(3,2) # normal/Gaussian distribution, mean 0 and unit variance

c = np.random.randn(10000)
print(c.mean(), c.var(), c.std())

d = np.random.randn(10, 3)
print(d.mean()) # mean of whole array: -0.1076827228882305

e = np.random.randint(3,10,size=(3,3)) # if we only pass one parameter, then
from 0-x
print(e)

f = np.random.choice(7, size=10)   # with integer is between 0 up to integer exclus
```

```
g = np.random.choice([1,2,3,4], size=8)
```

In [ ]:
```
# Eigenvalues
a = np.array([[1,2], [3,4]])   #Solving Linear Systems
eigenvalues, eigenvectors = np.linalg.eig(a)
print(eigenvalues)
print(eigenvectors) # column vectors

print(eigenvectors[:,0]) # column 0 corresponding to eigenvalue[0]

d = eigenvectors[:,0] * eigenvalues[0]  # verify: e-vec * e-val = A * e-vec
e = a @ eigenvectors[:, 0]
print(d, e) # [ 0.30697009 -0.21062466] [ 0.30697009 -0.21062466]

print(d == e) # [ True False] -> numerical issues # looks the same, but:

print(np.allclose(d,e)) # True # correct way to compare matrix

A = np.array([[1, 1], [1.5, 4]])  # -> 2 equations and 2 unknowns
b = np.array([2200,5050])

x = np.linalg.inv(A).dot(b) # Ax = b <=> x = A-1 b # But: inverse is slow and less

x = np.linalg.solve(A,b) # good
print(x) # [1500. 700.] # 1) load with np.loadtxt()

data = np.loadtxt('my_file.csv', delimiter=",",dtype=np.float32) # skiprows=1, ...
print(data.shape, data.dtype)

data = np.genfromtxt('my_file.csv', delimiter=",", dtype=np.float32)
print(data.shape)
```

In [ ]:
```
#.............Pandas___________
```

In [ ]:
```
#Loading data into Pandas
import pandas as pd

df = pd.read_csv('pokemon_data.csv')

# print(df.head(5))

# df_xlsx = pd.read_excel('pokemon_data.xlsx')
# print(df_xlsx.head(3))

# df = pd.read_csv('pokemon_data.txt', delimiter='\t')

# print(df.head(5)) - print first 5 rows from top
# print(df.tail(5)) - print first 5 rows from bottom

df['HP']
```

In [ ]:
```
#Reading Data in Pandas
#### Read Headers
df.columns
```

```
#print(df[['Name', 'Type 1', 'HP']])   ## Read each Column

#print(df.iloc[0:4])     ## Read Each Row

# for index, row in df.iterrows():
#     print(index, row['Name'])
#df.loc[df['Type 1'] == "Grass"]

#print(df.iloc[2,1])   ## Read a specific location (R,C)
```

In [ ]:
```
#Sorting/Describing Data
df.sort_values(['Type 1', 'HP'], ascending=[1,0])

df
```

In [ ]:
```
#Making changes to the data
#df['Total'] = df['HP'] + df['Attack'] + df['Defense'] + df['Sp. Atk'] + df['Sp. De

# df = df.drop(columns=['Total'])

df['Total'] = df.iloc[:, 4:10].sum(axis=1)

cols = list(df.columns)
df = df[cols[0:4] + [cols[-1]]+cols[4:12]]

df.head(5)
```

In [ ]:
```
#Saving our Data (Exporting into Desired Format)
# df.to_csv('modified.csv', index=False)

#df.to_excel('modified.xlsx', index=False)

df.to_csv('modified.txt', index=False, sep='\t')
```

In [ ]:
```
#Filtering Data
new_df = df.loc[(df['Type 1'] == 'Grass') & (df['Type 2'] == 'Poison') & (df['HP']

new_df.reset_index(drop=True, inplace=True)

new_df

new_df.to_csv('filtered.csv')
```

In [ ]:
```
#Conditional Changes
# df.loc[df['Total'] > 500, ['Generation','Legendary']] = ['Test 1', 'Test 2']

# df

df = pd.read_csv('modified.csv')

df
```

In [ ]:
```
#Aggregate Statistics (Groupby)
df = pd.read_csv('modified.csv')

df['count'] = 1

df.groupby(['Type 1', 'Type 2']).count()['count']
```

In [ ]:
```
#Working with large amounts of data
new_df = pd.DataFrame(columns=df.columns)

for df in pd.read_csv('modified.csv', chunksize=5):
    results = df.groupby(['Type 1']).count()

    new_df = pd.concat([new_df, results])
```

In [ ]:
```
pandas.errors: Custom exception and warnings classes that are raised by pandas.

pandas.plotting: Plotting public API.

pandas.testing: Functions that are useful for writing tests involving pandas object

pandas.api.extensions: Functions and classes for extending pandas objects.

pandas.api.indexers: Functions and classes for rolling window indexers.

pandas.api.interchange: DataFrame interchange protocol.

pandas.api.types: Datatype classes and functions.

pandas.api.typing: Classes that may be necessary for type-hinting.
```

In [27]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

x = [0,1,2,3,4] #Basic Graph
y = [0,2,4,6,8]

# Resize your Graph (dpi specifies pixels per inch. When saving probably should use
plt.figure(figsize=(8,5), dpi=100)

# Line 1

# Keyword Argument Notation
#plt.plot(x,y, label='2x', color='red', linewidth=2, marker='.', linestyle='--', ma

# Shorthand notation
# fmt = '[color][marker][line]'
plt.plot(x,y, 'b^--', label='2x')

## Line 2

# select interval we want to plot points at
x2 = np.arange(0,4.5,0.5)
```

```
# Plot part of the graph as line
plt.plot(x2[:6], x2[:6]**2, 'r', label='X^2')

# Plot remainder of graph as a dot
plt.plot(x2[5:], x2[5:]**2, 'r--')

# Add a title (specify font parameters with fontdict)
plt.title('Our First Graph!', fontdict={'fontname': 'Comic Sans MS', 'fontsize': 20

# X and Y labels
plt.xlabel('X Axis')
plt.ylabel('Y Axis')

# X, Y axis Tickmarks (scale of your graph)
plt.xticks([0,1,2,3,4,])
#plt.yticks([0,2,4,6,8,10])

# Add a legend
plt.legend()

# Save figure (dpi 300 is good when saving so graph has high resolution)
plt.savefig('mygraph.png', dpi=300)

# Show plot
plt.show()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[27], line 1
----> 1 import matplotlib.inline as plt
      2 import numpy as np
      3 import pandas as pd

ModuleNotFoundError: No module named 'matplotlib'
```

In [ ]:
```
abels = ['A', 'B', 'C'] #Bar Chart
values = [1,4,2]

plt.figure(figsize=(5,3), dpi=100)

bars = plt.bar(labels, values)

patterns = ['/', 'O', '*']
for bar in bars:
    bar.set_hatch(patterns.pop(0))

plt.savefig('barchart.png', dpi=300)

plt.show()
```

In [ ]:
```
gas = pd.read_csv('gas_prices.csv')  #Line Graph

plt.figure(figsize=(8,5))

plt.title('Gas Prices over Time (in USD)', fontdict={'fontweight':'bold', 'fontsize
```

```
plt.plot(gas.Year, gas.USA, 'b.-', label='United States')
plt.plot(gas.Year, gas.Canada, 'r.-')
plt.plot(gas.Year, gas['South Korea'], 'g.-')
plt.plot(gas.Year, gas.Australia, 'y.-')

# Another Way to plot many values!
# countries_to_look_at = ['Australia', 'USA', 'Canada', 'South Korea']
# for country in gas:
#     if country in countries_to_look_at:
#         plt.plot(gas.Year, gas[country], marker='.')

plt.xticks(gas.Year[::3].tolist()+[2011])

plt.xlabel('Year')
plt.ylabel('US Dollars')

plt.legend()

plt.savefig('Gas_price_figure.png', dpi=300)

plt.show()
```

In [ ]:
```
fifa = pd.read_csv('fifa_data.csv') #Load Fifa Data

fifa.head(5)
```

In [ ]:
```
bins = [40,50,60,70,80,90,100] #Histogram

plt.figure(figsize=(8,5))

plt.hist(fifa.Overall, bins=bins, color='#abcdef')

plt.xticks(bins)

plt.ylabel('Number of Players')
plt.xlabel('Skill Level')
plt.title('Distribution of Player Skills in FIFA 2018')

plt.savefig('histogram.png', dpi=300)

plt.show()
```

In [ ]:
```
left = fifa.loc[fifa['Preferred Foot'] == 'Left'].count()[0] #Pie Chart
right = fifa.loc[fifa['Preferred Foot'] == 'Right'].count()[0]

plt.figure(figsize=(8,5))

labels = ['Left', 'Right']
colors = ['#abcdef', '#aabbcc']

plt.pie([left, right], labels = labels, colors=colors, autopct='%.2f %%')

plt.title('Foot Preference of FIFA Players')
```

```
plt.show()
```

In [ ]:
```
plt.figure(figsize=(8,5), dpi=100) #Pie Chart #2

plt.style.use('ggplot')

fifa.Weight = [int(x.strip('lbs')) if type(x)==str else x for x in fifa.Weight]

light = fifa.loc[fifa.Weight < 125].count()[0]
light_medium = fifa[(fifa.Weight >= 125) & (fifa.Weight < 150)].count()[0]
medium = fifa[(fifa.Weight >= 150) & (fifa.Weight < 175)].count()[0]
medium_heavy = fifa[(fifa.Weight >= 175) & (fifa.Weight < 200)].count()[0]
heavy = fifa[fifa.Weight >= 200].count()[0]

weights = [light,light_medium, medium, medium_heavy, heavy]
label = ['under 125', '125-150', '150-175', '175-200', 'over 200']
explode = (.4,.2,0,0,.4)

plt.title('Weight of Professional Soccer Players (lbs)')

plt.pie(weights, labels=label, explode=explode, pctdistance=0.8,autopct='%.2f %%')
plt.show()
```

In [ ]:
```
plt.figure(figsize=(5,8), dpi=100) #Box and Whiskers Chart

plt.style.use('default')

barcelona = fifa.loc[fifa.Club == "FC Barcelona"]['Overall']
madrid = fifa.loc[fifa.Club == "Real Madrid"]['Overall']
revs = fifa.loc[fifa.Club == "New England Revolution"]['Overall']

#bp = plt.boxplot([barcelona, madrid, revs], labels=['a','b','c'], boxprops=dict(fa
bp = plt.boxplot([barcelona, madrid, revs], labels=['FC Barcelona','Real Madrid','N

plt.title('Professional Soccer Team Comparison')
plt.ylabel('FIFA Overall Rating')

for box in bp['boxes']:
    # change outline color
    box.set(color='#4286f4', linewidth=2)
    # change fill color
    box.set(facecolor = '#e0e0e0' )
    # change hatch
    #box.set(hatch = '/')

plt.show()
```

In [ ]:
```
#.... TK inter
# some widgets
# Label, Button, checkbutton,radiobutton, listbox, scrollbar, Menu,Enty
import tkinter as tk
def say_hello():
    print("Hello from the button!")
root = tk.Tk()
```

```
root.title("Simple Tkinter Example")
label = tk.Label(root, text="Welcome!")
label.pack(pady=10) # Add some padding
button = tk.Button(root, text="Greet", command=say_hello)
button.pack()
root.mainloop()
```

In [ ]:

```
import tkinter as tk
w = tk.Tk()
w.geometry("1000x500") # Set window size

w.title("Quiz")
heading = tk.Label(w, text="Quiz", font=("Arial", 20, "bold"))
heading.grid(row=1, column=2, pady=20)

tk.Label(w, text="Name").grid(row=4, column=0,sticky="E")
entry1 = tk.Entry(w)
entry1.grid(row=4, column=1)
tk.Label(w, text="Phonenumber").grid(row=5, column=0,sticky="E")
entry2 = tk.Entry(w)
entry2.grid(row=5, column=1)
tk.Label(w, text="Email").grid(row=6, column=0,sticky="E")
entry2 = tk.Entry(w)
entry2.grid(row=6, column=1)


questions = ["What is 2+2?", "What is the capital of India?", "Who created Python?"
index = 0

def next_question():
    global index
    if index < len(questions):
        question_label.config(text=questions[index])
        index += 1
        selected_option = tk.StringVar(value="Option1")
        radio1 = tk.Radiobutton(w, text="Option 1", variable=selected_option, value
        radio1.grid(row=10, column=1, sticky="w")
        radio2 = tk.Radiobutton(w, text="Option 2", variable=selected_option, value
        radio2.grid(row=11, column=1, sticky="w")
        if index == 0:
            print('A',index)
            pass
        elif index == (len(questions)):
            print('B',index)
            pass
        else:
            print('C',index)
            back_button = tk.Button(w, text="Back", command=go_back)
            back_button.grid(row=15, column=5,pady=20)
        # index += 1
        if index == len(questions):
            next_button.config(text="Submit", command=submit_quiz)
    else:
        submit_quiz()

def submit_quiz():
```

```
    question_label.config(text="Quiz Submitted!")
    next_button.config(state="disabled")
question_label = tk.Label(w, text="", font=("Arial", 14))
question_label.grid(row=7, column=2,sticky="w",pady=20)
next_button = tk.Button(w, text="Next", command=next_question)
next_button.grid(row=15, column=6,sticky="e", pady=20)

def go_back():
    if index > 0:
        show_page(index - 1)

classes = [
    "1st", "2nd", "3rd", "4th", "5th", "6th",
    "7th", "8th", "9th", "10th", "1st PUC", "2nd PUC"
]
def show_selected():
    selected = listbox.get(listbox.curselection())
    print("Selected:", selected)
tk.Label(w, text="Select your class:", font=("Arial", 14)).grid(row=1, column=3, pa
listbox = tk.Listbox(w, height=12)
for item in classes:
    listbox.insert(tk.END, item)
listbox.grid(row=1, column=3, pady=20)
selbutton = tk.Button(w, text="Show Selected", command=show_selected).grid(row=1, c


w.mainloop()
```

In [ ]:
```
# ..... SQL Data Base
```

In [ ]:
```
import pymysql
from pymysql import Error

def fetch_all():
    query = '''
    select * from books
    '''
    cur.execute(query)
    book = cur.fetchall()
    return book

def display_book():
    print(fetch_all())

def delete_bookid():
    bid1 = int(input('Please enter bid to delete: '))
    query = '''
    delete from books where bid=%s
    '''
    value = (bid1,)
    x=cur.execute(query, value)
    print(x)
    conn.commit()

def update_bookid():
```

```
    bid1 = int(input('Please enter bid to update: '))
    book = fetch_all()
    found = 0
    for book in book:
        if book[0] == bid1:
            found = 1
            print('Existing details of book',book)
            aname = input('Enter Author name: ')
            pdate = input('Enter publish date(YYYYMMDD): ')
            npage = int(input('Enter number of pages: '))
            query = '''
            UPDATE BOOKS SET aname=%s,pdate=%s,npage=%s WHERE bid=%s
            '''
            values = (aname,pdate,npage,bid1)
            cur.execute(query, values)
            conn.commit()
    if found == 0:
        print('Not found in datase')

def newentry_bookid():
    book = fetch_all()
    print('Existing details of book \n',book)
    bid = int(input('Enter Book id number: '))
    aname = input('Enter Author name: ')
    pdate = input('Enter publish date(YYYYMMDD): ')
    npage = int(input('Enter number of pages: '))
    query = '''
    insert into BOOKS (bid,aname,pdate,npage)
    values(%s,%s,%s,%s)
    '''
    values = (bid,aname,pdate,npage,)
    cur.execute(query, values)
    conn.commit()

try:
    conn = pymysql.connect(
        host='localhost',
        user='root',
        password='password',
        database='books'
    )
    cur = conn.cursor()
    # if conn.is_connected():
    print('_____________________')
    while(1):
        print('1. Display Entire book')
        print('2. Delete book ID')
        print('3. Update book ID')
        print('4. Add new entry book ID')
        print('5. Exit')
        option = int(input('Enter an option: '))
        if option == 1:
            display_book()
        elif option == 2:
            delete_bookid()
        elif option == 3:
```

```
            update_bookid()
        elif option == 4:
            newentry_bookid()
        else:
            exit()
    # else:
    #     print('???????????????')
    cur.close()

finally:
    conn.close()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: