

## 1. Explain module in python.

Module is same as a code library.

A file containing a set of functions we want to include in our application.

### Create a Module

To create a module just save the code in a file with the file extension `.py`:

### Example

Save this code in a file named `mymodule.py`

```
def greeting(name):  
    print("Hello, " + name)
```

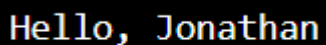
### Use a Module

We can use the module we just created, by using the `import` statement:

### Example

Import the module named `mymodule`, and call the `greeting` function:

```
import mymodule  
  
mymodule.greeting("Jonathan")
```

A black rectangular box containing the text "Hello, Jonathan" in a yellow, monospaced font.

**Note:** When using a function from a module, use the syntax: *module\_name.function\_name*.

## 2. How to create pie chart using matplotlib

With Pyplot, you can use the `pie()` function to draw pie charts:

Add labels to the pie chart with the `labels` parameter.

The `labels` parameter must be an array with one label for each wedge:

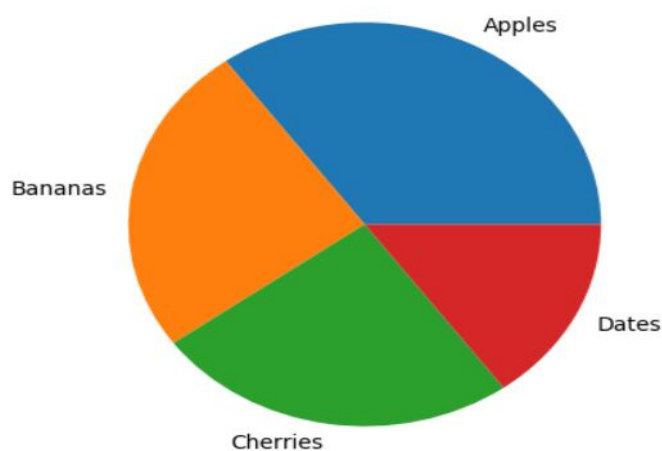
### Example

A simple pie chart:

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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

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



### 3. How to create bars using matplotlib

With Pyplot, you can use the `bar()` function to draw bar graphs:

The `bar()` function takes arguments that describes the layout of the bars.

The categories and their values represented by the *first* and *second* argument as arrays.

The keyword argument `color` to set the color of the bars:

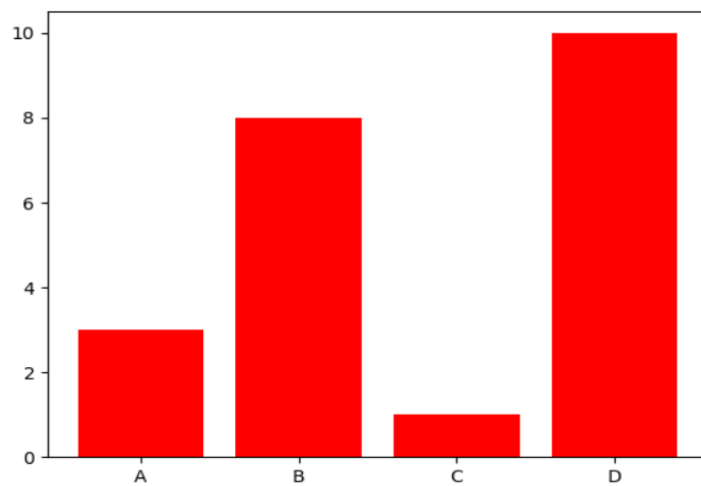
#### Example

Draw 4 red bars:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color = "red")
plt.show()
```



#### 4. How to create Subplot using matplotlib

The `subplot()` function takes three arguments that describes the layout of the figure.

The layout is organized in rows and columns, which are represented by the *first* and *second* argument.

The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
```

```
plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
```

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

```
#plot 1:
```

```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
```

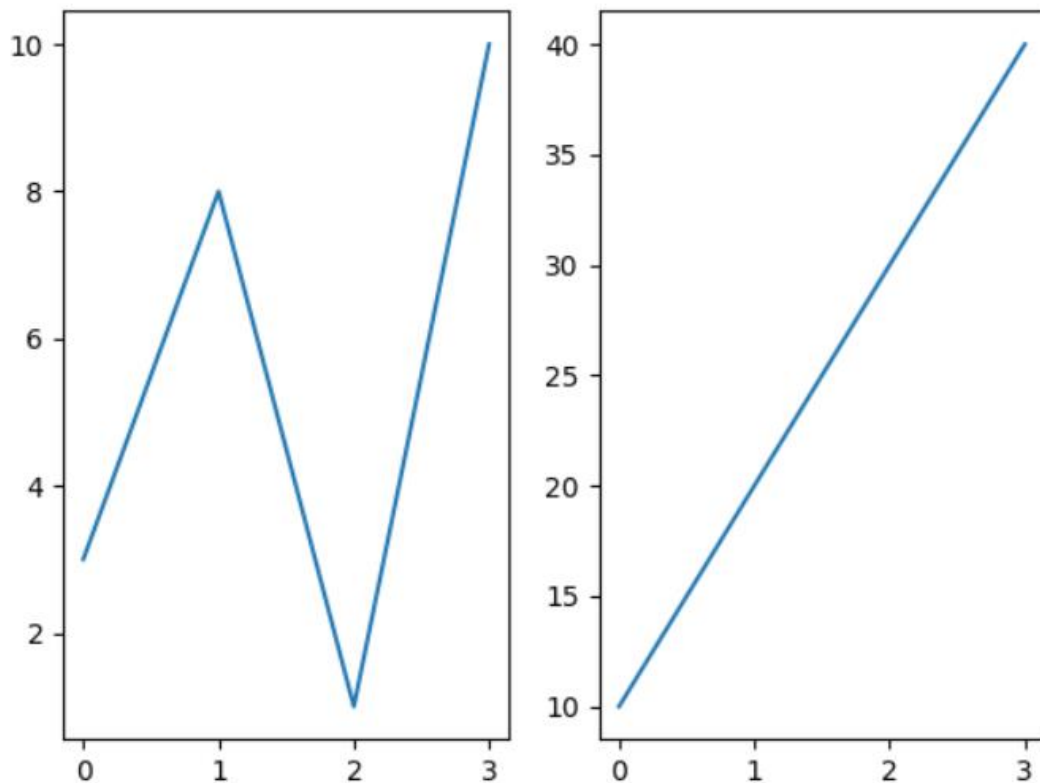
```
plt.subplot(1, 2, 1)
plt.plot(x,y)
```

```
#plot 2:
```

```
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
```

```
plt.subplot(1, 2, 2)
plt.plot(x,y)
```

```
plt.show()
```



## 5. Explain opening a file with syntax and example.

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:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

### Syntax

To open a file for reading it is enough to specify the name of the file:

```
f = open("demofile.txt")
```

The code above is the same as:

```
f = open("demofile.txt", "rt")
```

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

**Note:** Make sure the file exists, or else you will get an error.

## 6. Explain closing the file with syntax and example.

### Close Files

It is a good practice to always close the file when you are done with it.

### Example

Close the file when you are finish with it:

```
f = open("demofile.txt", "r")  
print(f.readline())  
f.close()
```

```
Hello! Welcome to demofile.txt
```

**Note:** You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

## 7. How to write to an file?

To write to an existing file, you must add a parameter to the `open()` function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

### Example

Open the file "demofile2.txt" and append content to the file:

---

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

### Example

Open the file "demofile3.txt" and overwrite the content:

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

```
Woops! I have deleted the content!
```

**Note:** the "w" method will overwrite the entire file.

## 8. Explain

- a) Create a new file
- b) Delete a file

### Create

To create a new file in Python, use the `open()` method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

### Example

Create a file called "myfile.txt":

```
f = open("myfile.txt", "x")
```

Result: a new empty file is created!

### Example

Create a new file if it does not exist:

```
f = open("myfile.txt", "w")
```

### Delete

To delete a file, you must import the OS module, and run its `os.remove()` function:

### Example

Remove the file "demofile.txt":

```
import os  
os.remove("demofile.txt")
```

## 9. What is data frame? How to create a data frame?

A Pandas Data Frame is a 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns.

### Example

Create a simple Pandas Data Frame:



```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

	calories	duration
0	420	50
1	380	40
2	390	45

## 10.What are the two methods for viewing the data of data frame?

One of the most used method for getting a quick overview of the DataFrame, is the `head()` method.

The `head()` method returns the headers and a specified number of rows, starting from the top.

### Example

Get a quick overview by printing the first 10 rows of the DataFrame:

```
import pandas as pd

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

print(df.head(10))
```

**Note:** if the number of rows is not specified, the `head()` method will return the top 5 rows.

There is also a `tail()` method for viewing the *last* rows of the DataFrame.

The `tail()` method returns the headers and a specified number of rows, starting from the bottom.

### Example

Print the last 5 rows of the DataFrame:

```
print(df.tail())
```

	Duration	Pulse	Maxpulse	Calories
164	60	105	140	290.8
165	60	110	145	300.4
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

## 11.How to create a 1-D array and access the elements of it?

### 1-D Arrays

An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.

These are the most common and basic arrays.

#### Example

Create a 1-D array containing the values 1,2,3,4,5:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

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

### Access Array Elements

Array indexing is the same as accessing an array element.

We can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

#### Example

Get the first element from the above array:

```
print(arr[0])
```

1

## 12.Explain slicing of arrays.

### Slicing arrays

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: *[start:end]*.

If we don't pass start its considered 0

If we don't pass end its considered length of array in that dimension

### Example

Slice elements from index 1 to index 5 from the following array:

```
import numpy as np

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

print(arr[1:5])
```

```
[2 3 4 5]
```

## Multiple choice questions.

1. How do you import a module named "example\_module" in Python?
  - A) **import example\_module**
  - B) include example\_module
  - C) load example\_module
  - D) use example\_module
2. Which statement accurately describes the import statement in Python?
  - A) **It loads the module only once, even if imported multiple times in a program.**
  - B) It loads the module every time it is imported in a program.
  - C) It is used to export functions and variables from a module.
  - D) It imports all functions and variables from a module automatically.
3. How do you access a function named "my\_function" from an imported module named "my\_module"?
  - A) my\_function.my\_module
  - B) **my\_module.my\_function**
  - C) my\_function->my\_module

- D) `my_module::my_function`
4. What happens when you import a module in Python?
- A) **It executes the module's code.**
  - B) It makes the module available for use in the current program.
  - C) It prints the module's documentation.
  - D) It initializes the module's variables.
5. How can you alias a module named "long\_module\_name" to use it with a shorter name?
- A) `alias long_module_name as lm`
  - B) `from long_module_name import as lm`
  - C) **`import long_module_name as lm`**
  - D) `use long_module_name as lm`
6. How can you create a module in Python?
- A) **Write Python code in a text file with a `.py` extension.**
  - B) Compile Python code into a shared library.
  - C) Use a built-in Python function to generate a module.
  - D) Modules can only be created by Python core developer
7. What is NumPy?
- a) **A scientific computing library in Python**
  - b) A machine learning framework
  - c) A data visualization tool
  - d) A web development framework
8. Which of the following is the correct way to import NumPy in Python?
- a) **`import numpy as np`**
  - b) `from numpy import *`
  - c) `import numpy`
  - d) `from numpy import numpy`
9. What is the primary data structure in NumPy?
- a) **Array**
  - b) List
  - c) Dictionary
  - d) Set
10. What function is used to generate random numbers in NumPy?
- a) `random.random()`
  - b) `np.rand()`
  - c) **`np.random()`**
  - d) `random.randint()`
11. What is a DataFrame in Python Pandas?
- a) A graphical representation of data
  - b) **A two-dimensional labeled data structure**
  - c) A function for plotting data
  - d) A machine learning model
12. Which library in Python is commonly used for working with DataFrames?
- a) NumPy
  - b) **Pandas**
  - c) Matplotlib
  - d) Scikit-learn
13. How do you create a DataFrame from a dictionary `data` in Pandas?
- a) **`pd.DataFrame(data)`**
  - b) `DataFrame(data)`

- c) `pd.createDataFrame(data)`
  - d) `DataFrame.from_dict(data)`
14. What function is used to display the first few rows of a DataFrame in Pandas?
- a) `display()`
  - b) `show()`
  - c) **`head()`**
  - d) `top()`
15. How can you access a specific column named 'column\_name' in a DataFrame `df`?
- a) `df.column_name`
  - b) `df.get_column('column_name')`
  - c) **`df['column_name']`**
  - d) `df.column['column_name']`
16. What is Matplotlib?
- a) A machine learning library
  - b) A data manipulation library
  - c) **A data visualization library**
  - d) A web development framework
17. How do you import Matplotlib in Python?
- a) `import matplotlib`
  - b) **`import matplotlib.pyplot as plt`**
  - c) `import matplotlib as mpl`
  - d) `import plt.matplotlib`
18. Which of the following is the correct way to create a basic line plot using Matplotlib?
- a) **`plt.plot()`**
  - b) `plt.line()`
  - c) `plt.create_plot()`
  - d) `plt.line_plot()`
19. What function is used to add a title to a Matplotlib plot?
- a) `plt.add_title()`
  - b) **`plt.title()`**
  - c) `plt.plot_title()`
  - d) `plt.set_title()`
20. How can you add labels to the x-axis and y-axis in a Matplotlib plot?
- a) `plt.add_labels()`
  - b) **`plt.x_label()` and `plt.y_label()`**
  - c) `plt.xlabel()` and `plt.ylabel()`
  - d) `plt.set_xlabel()` and `plt.set_ylabel()`
21. What function is used to create a scatter plot in Matplotlib?
- a) **`plt.scatter()`**
  - b) `plt.plot()`
  - c) `plt.create_scatter()`
  - d) `plt.scatter_plot()`
22. How do you create subplots in Matplotlib?
- a) `plt.create_subplots()`
  - b) `plt.add_subplot()`
  - c) `plt.subplot()`
  - d) **`plt.subplots()`**
23. How can you add a legend to a Matplotlib plot?
- a) **`plt.legend()`**
  - b) `plt.add_legend()`

- c) plt.create\_legend()
  - d) plt.set\_legend()
24. How do you create a histogram in Matplotlib?
- a) **plt.hist()**
  - b) plt.create\_histogram()
  - c) plt.plot\_hist()
  - d) plt.histogram()
25. Which method is used to add grid lines to a Matplotlib plot?
- a) **plt.grid()**
  - b) plt.add\_grid()
  - c) plt.create\_grid()
  - d) plt.set\_grid()