ASSIGNMENT

Name: Anirban Majumder

Course:

Decode Data Science With Machine Learning

Date: 27th August, 2024

PW SKILLS ASSIGNMENT

DECODE DATA SCIENCE WITH MACHINE LEARNING

Question No. – 1:

1.1 Write the answer to these questions:

Note: Give at least one example for each of the questions.

- What is the difference between static and dynamic variables in Python?

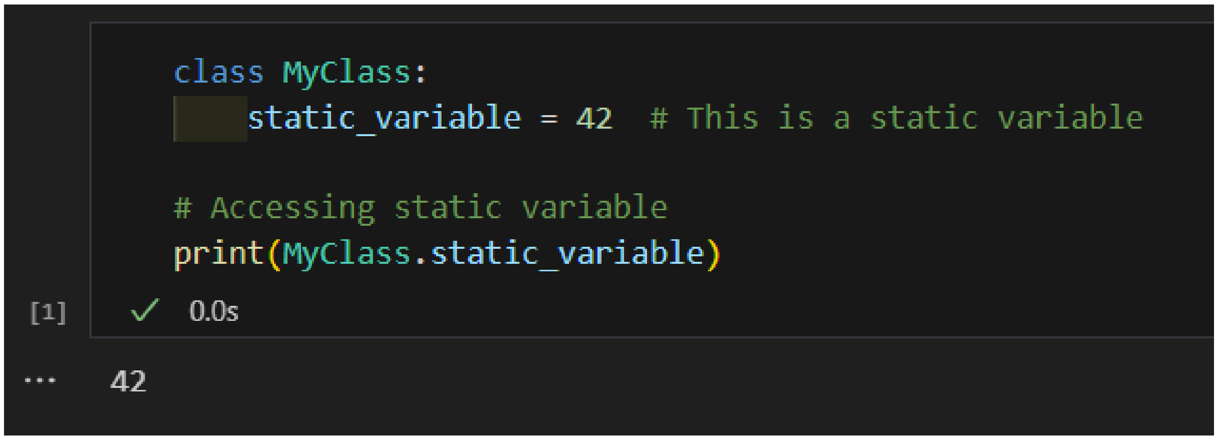
Answer:

The key differences between Static and Dynamic Variables in Python are as follows:-

Features or Parameters	***Static Variable***	***Dynamic Variable***
Scope	Classes are typically connected to static variables. They are not included in any instance methods, although they are specified inside a class. Every instance of the class shares these variables.	Generally, instance variables are dynamic variables. They are linked to a particular instance of a class and defined within methods (often __init__).
Initialization	Static variables are initialized only once, generally at the time of defining a class.	Whenever an instance of a class is created, dynamic variables are initialized.
Lifetime	Static variables are retained by the program for its entire duration.	Dynamic variables have a lifespan that is correlated with the instance to which they belong.
Access	The class name or an instance of the class can be used to access static variables.	Only instances of the class have access to dynamic variables.
Usage	For constants or variables that need to be shared by all instances of the class, static variables are useful.	When it comes to properties that are specific to each instance, dynamic variables are useful.

Examples:

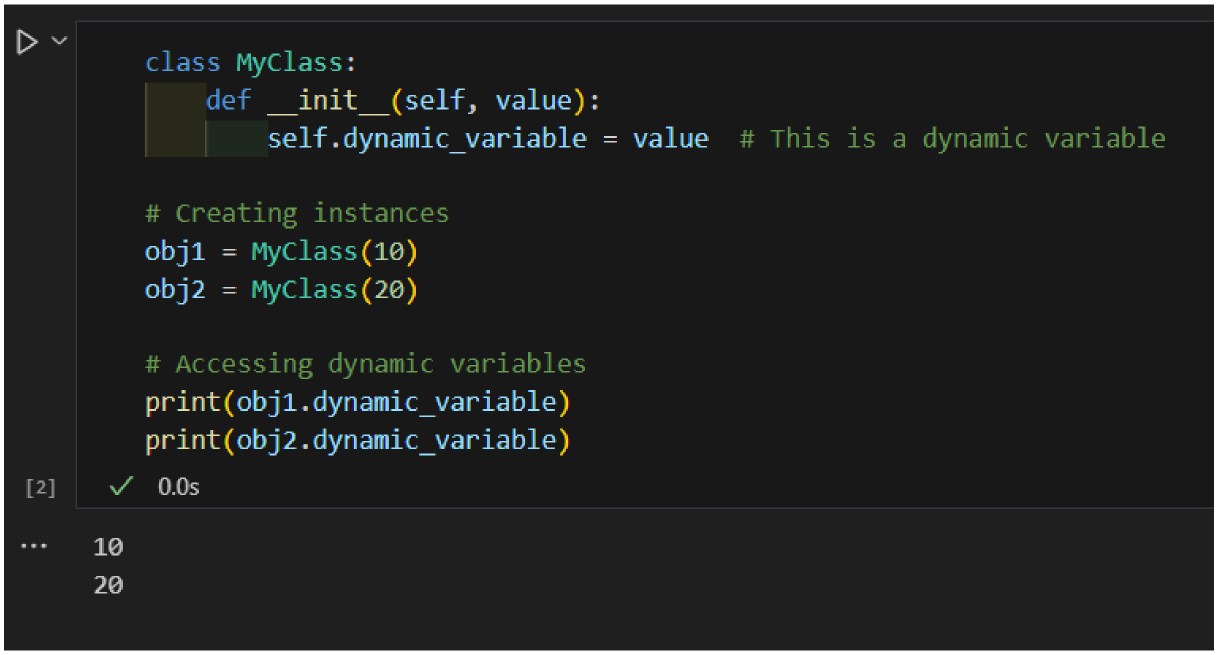
Code Snippet of Static Variable:



```
class MyClass:
    static_variable = 42  # This is a static variable

# Accessing static variable
print(MyClass.static_variable)
```

```
[1] ✓ 0.0s
... 42
```

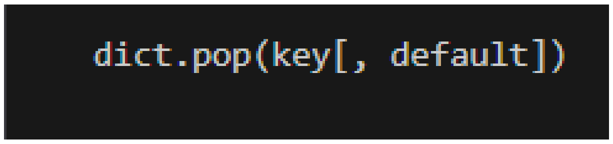
Code Snippet of Dynamic Variable:



```
class MyClass:
    def __init__(self, value):
        self.dynamic_variable = value  # This is a dynamic variable

# Creating instances
obj1 = MyClass(10)
obj2 = MyClass(20)

# Accessing dynamic variables
print(obj1.dynamic_variable)
print(obj2.dynamic_variable)
```

```
[2] ✓ 0.0s
... 10
    20
```

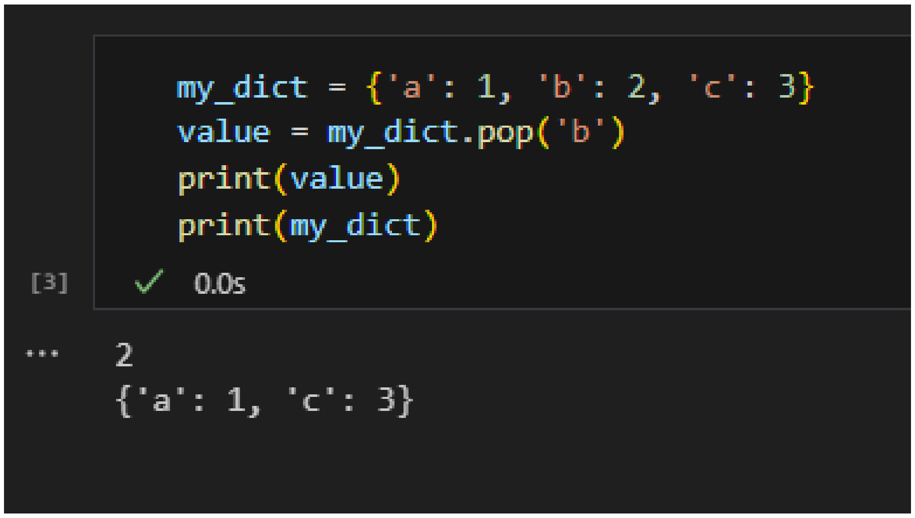
- Explain the purpose of "pop","popitem","clear()" in a dictionary with suitable examples.
  Answer:
  The purpose of "pop ()", "popitem ()" and "clear ()" in a dictionary, with suitable examples are as follows:-
  ➢ **"pop ()":** The "pop ()" method removes a specified key from the dictionary and returns the corresponding value. If the key is not found, a 'KeyError' is raised unless a default value is provided.
  ✓ **Syntax:**



```
dict.pop(key[, default])
```

where,
'key' → refers to the key to be removed.
'default' → this is optional and is used to return a value, if the key is not found. If this is not provided and the key is not found, a KeyError is raised.

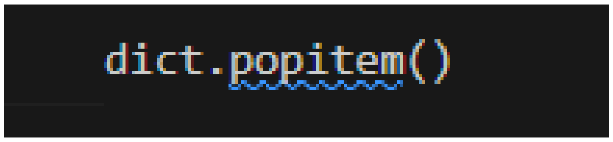
✓ **Example:**



```
my_dict = {'a': 1, 'b': 2, 'c': 3}
value = my_dict.pop('b')
print(value)
print(my_dict)
```

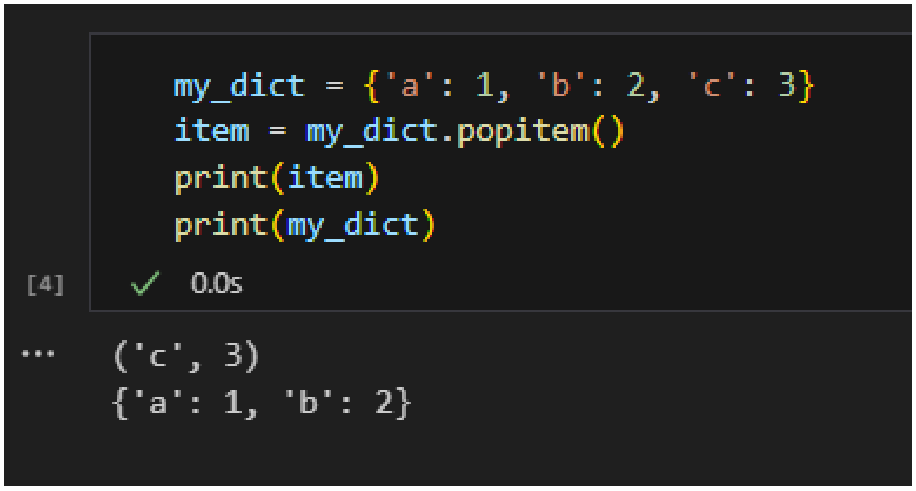
```
[3] ✓ 0.0s
2
{'a': 1, 'c': 3}
```

➢ **"popitem ()":** The "popitem ()" method removes and returns an arbitrary (key, value) pair from the dictionary. It is often used to destructively iterate over a dictionary. If the dictionary is empty, a 'KeyError' is raised.
✓ **Syntax:**



```
dict.popitem()
```

- ✓ **Example:**

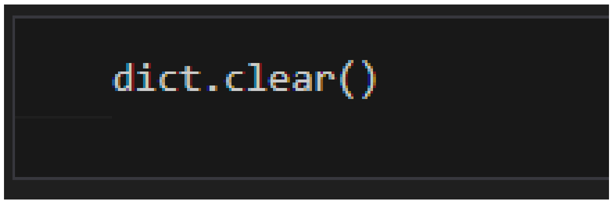


```
my_dict = {'a': 1, 'b': 2, 'c': 3}
item = my_dict.popitem()
print(item)
print(my_dict)
```

```
('c', 3)
{'a': 1, 'b': 2}
```

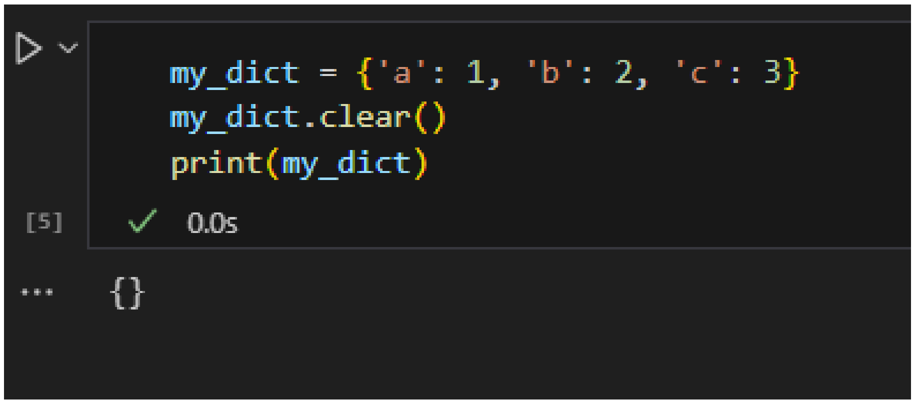
➢ **<u>"clear ()":</u>** The "clear ()" method removes all items from the dictionary, effectively making it empty.

- ✓ **Syntax:**



```
dict.clear()
```

- ✓ **Example:**



```
my_dict = {'a': 1, 'b': 2, 'c': 3}
my_dict.clear()
print(my_dict)
```

```
{}
```

- What do you mean by FrozenSet? Explain it with suitable examples

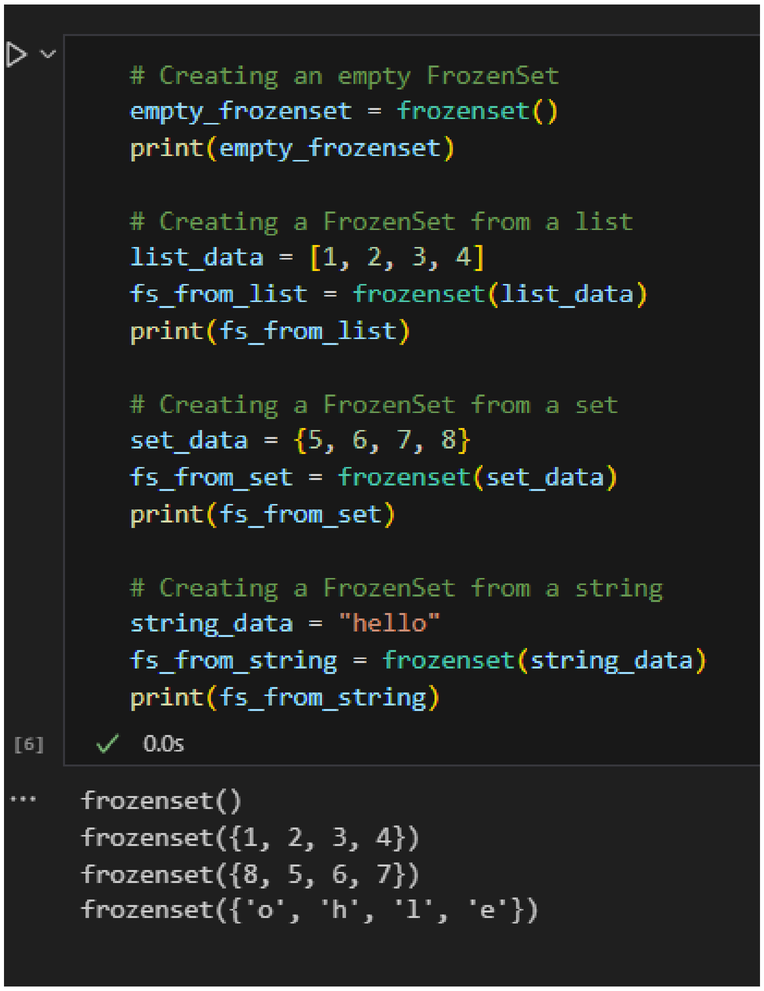
  Answer:

  An immutable form of a set is called a FrozenSet in Python. A FrozenSet's elements cannot be changed, added, or removed after it has been formed. Because of its immutability, FrozenSet can be used as an element of another set or as a key in a dictionary.

  Characteristics of a FrozenSet: The characteristics of a FrozenSet are as follows:-

  - It is an immutable set in Python.
  - Since it is immutable, it is hashable, signifying that it can be hashed, allowing it to be used as a dictionary key or a set element.
  - It is iterable.
  - It supports common set operations like union, intersection, difference, and symmetric difference, but does not support methods that modify the set (like add, remove, etc.).

  Example:



```
# Creating an empty FrozenSet
empty_frozenset = frozenset()
print(empty_frozenset)

# Creating a FrozenSet from a list
list_data = [1, 2, 3, 4]
fs_from_list = frozenset(list_data)
print(fs_from_list)

# Creating a FrozenSet from a set
set_data = {5, 6, 7, 8}
fs_from_set = frozenset(set_data)
print(fs_from_set)

# Creating a FrozenSet from a string
string_data = "hello"
fs_from_string = frozenset(string_data)
print(fs_from_string)
```

[6] ✓ 0.0s

```
frozenset()
frozenset({1, 2, 3, 4})
frozenset({8, 5, 6, 7})
frozenset({'o', 'h', 'l', 'e'})
```

- Differentiate between mutable and immutable data types in Python and give examples of mutable and immutable data types.

  Answer:

  The difference between Mutable and Immutable Data Types in Python along with their examples are as follows:-

Features or Parameters	***Mutable Data Type***	***Immutable Data Type***
Definition	Mutable data types are those whose values can be changed in place after they are created, which means they can be modified without creating a new object.	Immutable data types are those whose values cannot be changed after they are created and any modification results in the creation of a new object.
Characteristics	These can be changed after their creation, i.e., elements can be added, removed or modified and the object's identity (its memory address) does not change when its content changes.	These cannot be changed after their creation, and any modification results in the creation of a new object.
Example	List, Dictionary, Set.	String, Tuple, FrozenSet.

- What is __init__? Explain with an example.

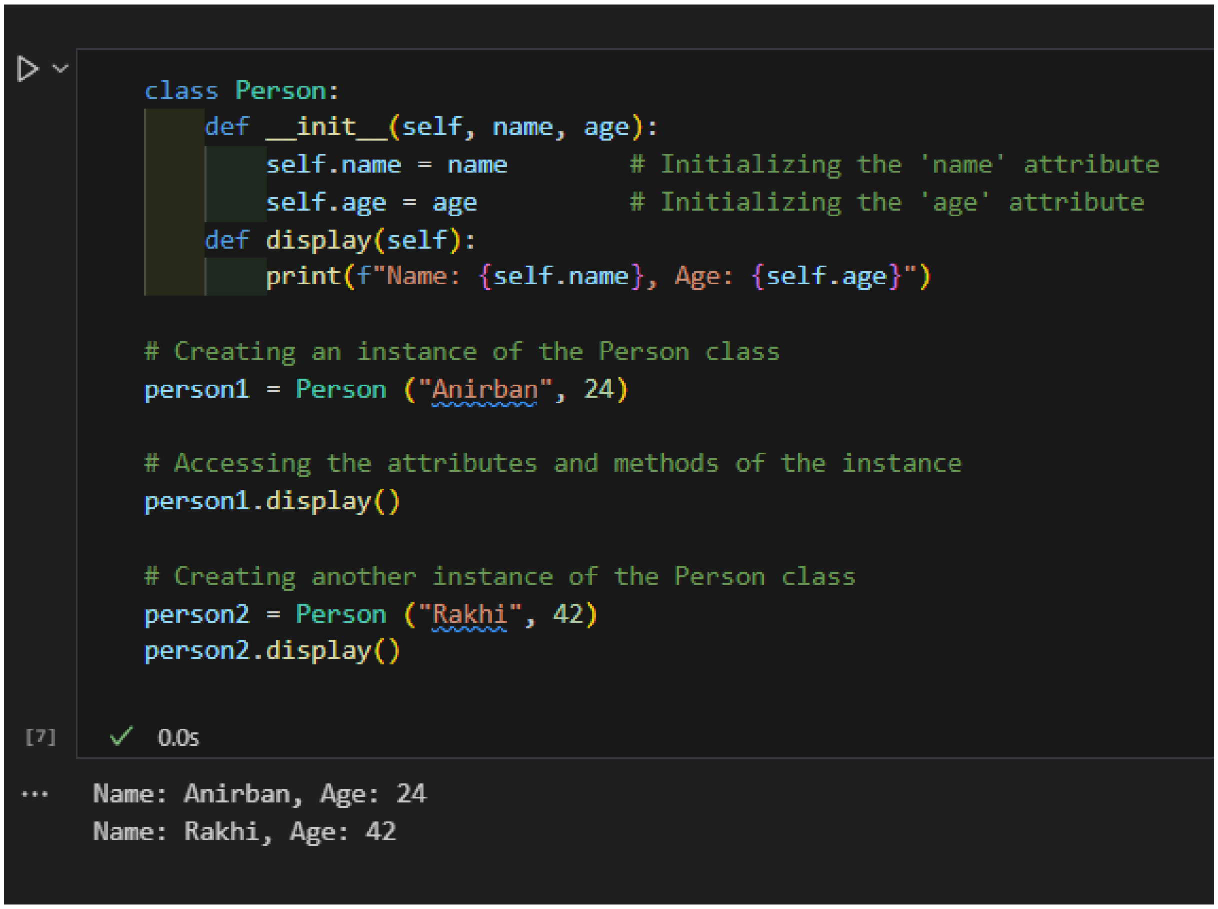
  Answer:

  The "__init__" method in Python is a special method that is called when an instance (object) of a class is created. It is commonly referred to as the "constructor" because it initializes the object's attributes.

  Characteristics of "__init__":

  - ***Initialization:*** It initializes the state of an object by assigning values to the instance variables.
  - ***Self Parameter:*** The first parameter of "__init__" is always 'self', which refers to the instance being created.
  - ***Pseudo Constructor:*** The "__init__" method doesn't create the object, i.e., it is not a true constructor but is used to initialize the attributes of the objects after it has been created.
  - The __init__ method is essential for setting up an object's initial state and ensuring that necessary attributes are properly initialized when an instance is created.

Example:



```
class Person:
    def __init__(self, name, age):
        self.name = name          # Initializing the 'name' attribute
        self.age = age            # Initializing the 'age' attribute
    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

# Creating an instance of the Person class
person1 = Person ("Anirban", 24)

# Accessing the attributes and methods of the instance
person1.display()

# Creating another instance of the Person class
person2 = Person ("Rakhi", 42)
person2.display()
```

[7] ✓ 0.0s

```
Name: Anirban, Age: 24
Name: Rakhi, Age: 42
```

- What is docstring in Python? Explain with an example.

  Answer:

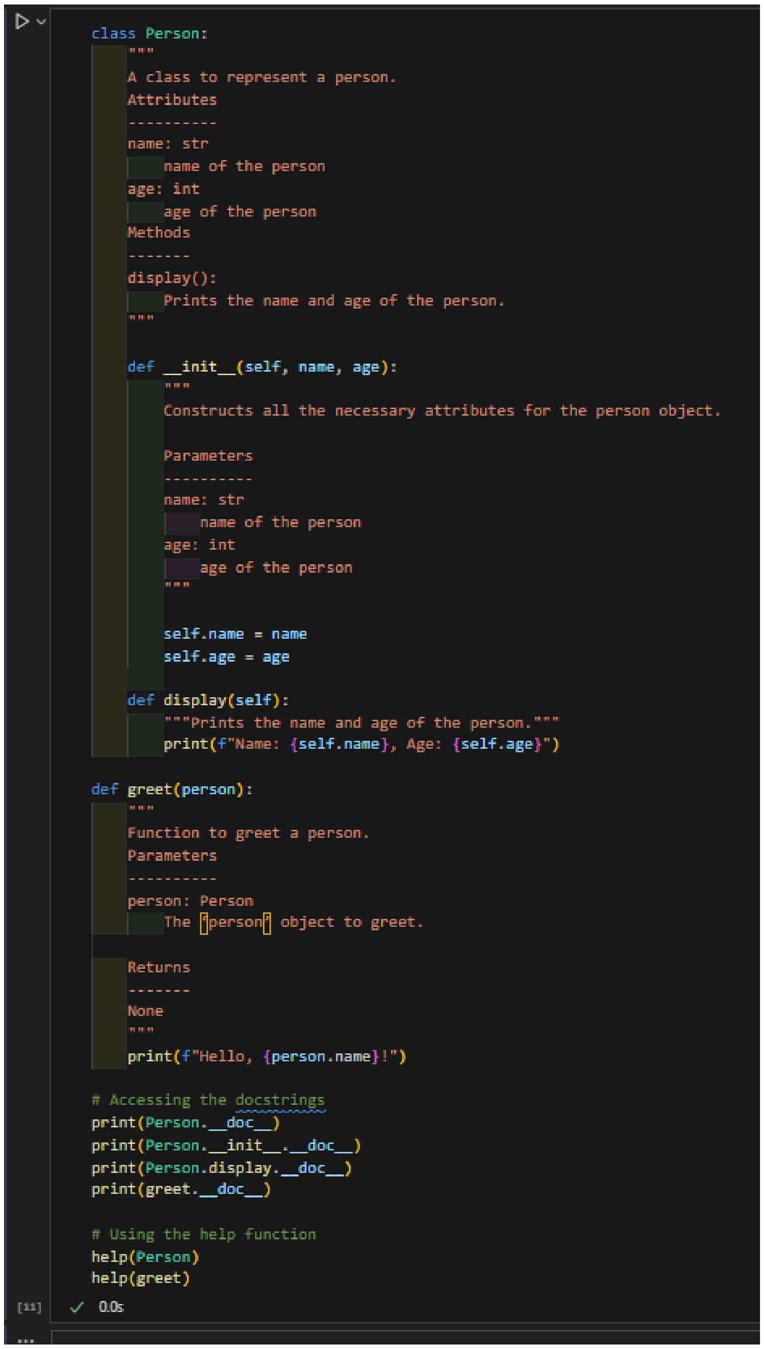
  A unique type of string called a docstring is used in Python to describe a module, class, method, or function. It comes after the definition of the item it documents and is surrounded by triple quotes (‘’’ or """). Docstrings, which are accessible by the “help()” method or the “__doc__” property, offer a practical means of linking documentation to Python code.

  Characteristics of Docstrings:

- ***Placement:*** Placed directly below the function, method, class or module definition.
- ***Format:*** Docstrings are enclosed within triple quotes, either single (‘’’) or double (“””).
- ***Accessibility:*** Docstrings can be accessed using the “__doc__” attribute or the “help()” function.

- Docstrings are an essential part of writing readable and maintainable code as they provide a built-in way to document the purpose and usage of modules, classes, and functions.

  Example:



```python
class Person:
    """
    A class to represent a person.
    Attributes
    ----------
    name: str
        name of the person
    age: int
        age of the person
    Methods
    -------
    display():
        Prints the name and age of the person.
    """

    def __init__(self, name, age):
        """
        Constructs all the necessary attributes for the person object.

        Parameters
        ----------
        name: str
            name of the person
        age: int
            age of the person
        """

        self.name = name
        self.age = age

    def display(self):
        """Prints the name and age of the person."""
        print(f"Name: {self.name}, Age: {self.age}")

def greet(person):
    """
    Function to greet a person.
    Parameters
    ----------
    person: Person
        The 'person' object to greet.

    Returns
    -------
    None
    """
    print(f"Hello, {person.name}!")

# Accessing the docstrings
print(Person.__doc__)
print(Person.__init__.__doc__)
print(Person.display.__doc__)
print(greet.__doc__)

# Using the help function
help(Person)
help(greet)
```

- What are unit tests in Python?

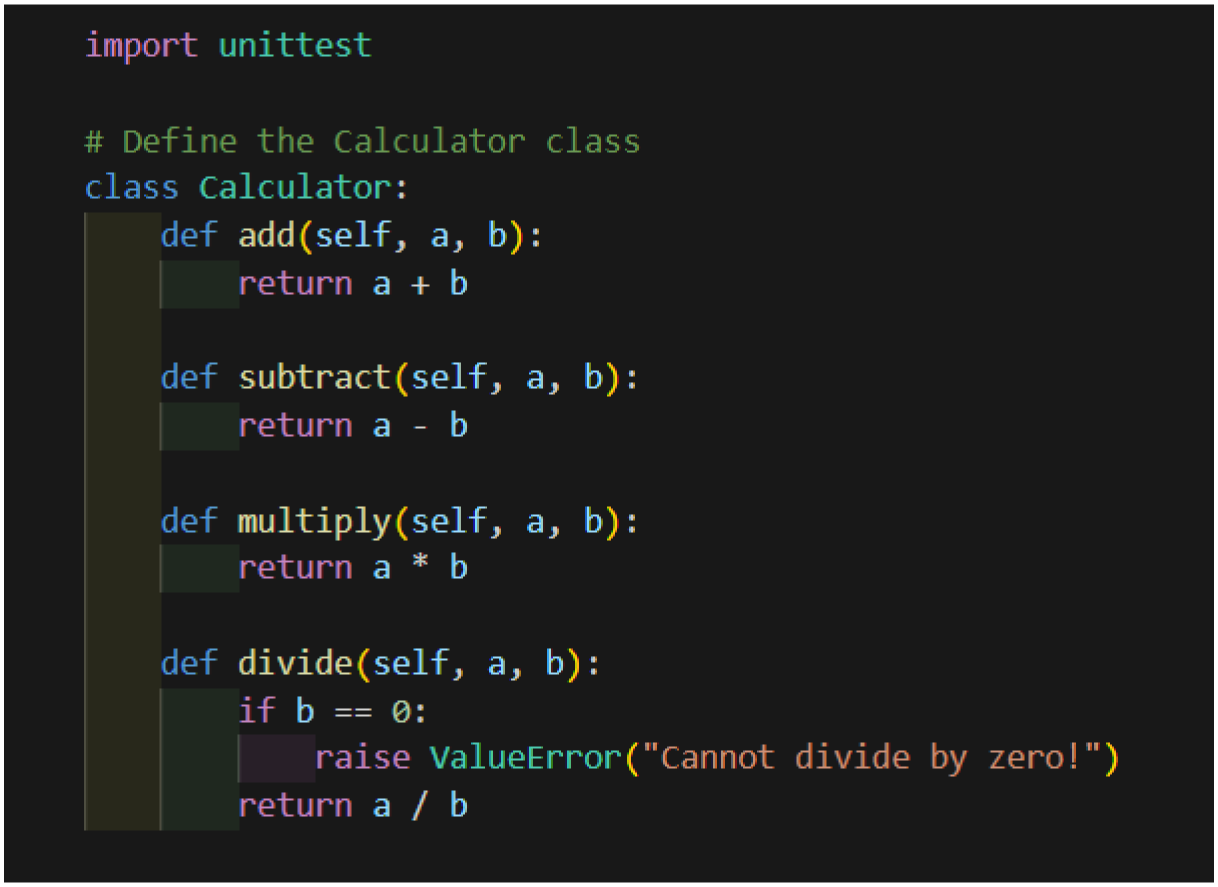
  Answer:

  Python unit tests are a means to test discrete code units to make sure every component performs as intended. Verifying that each software unit operates as intended is the main objective of unit testing. The "unittest" module from the Python Standard Library is commonly used when writing unit tests in Python.

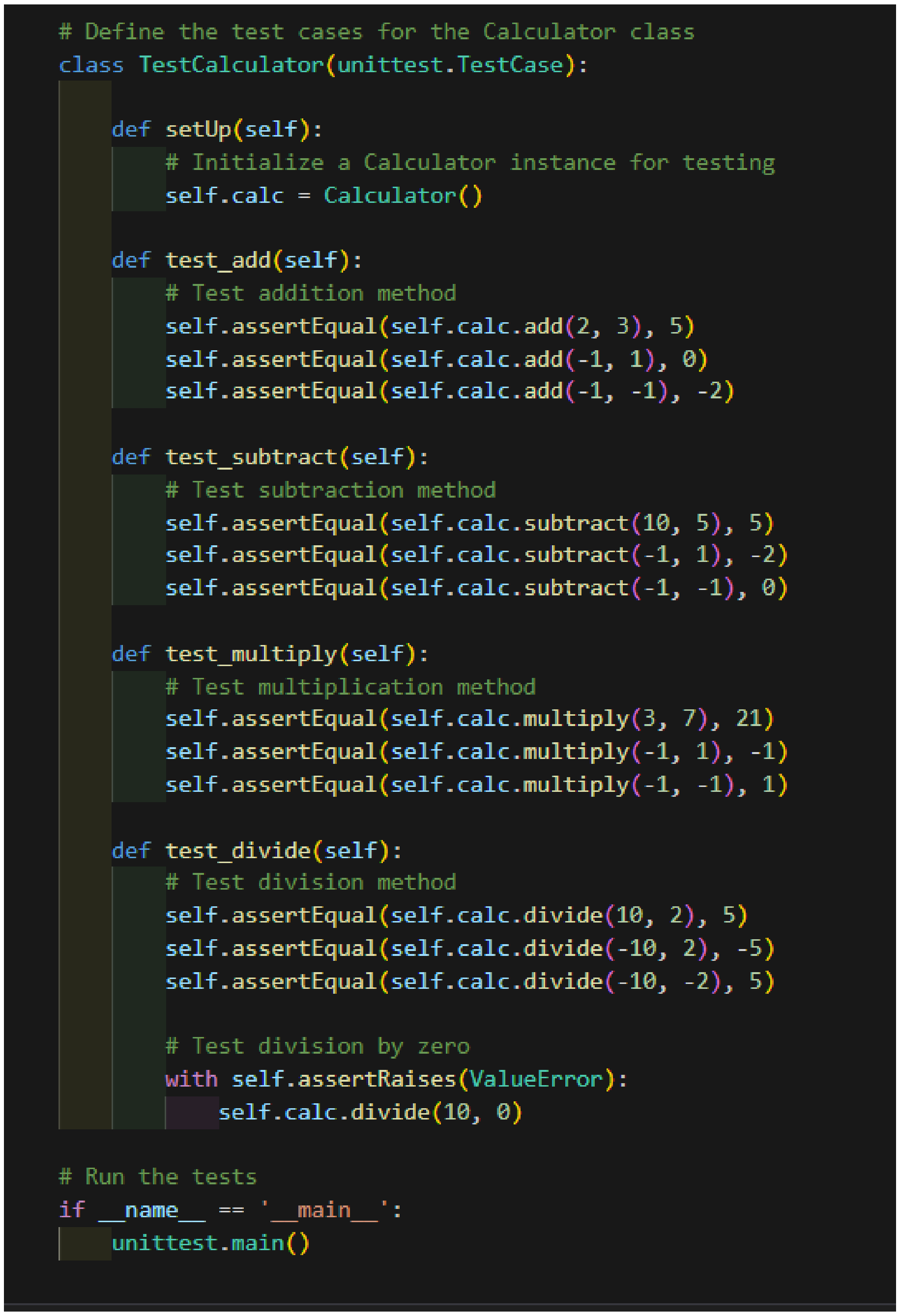
  Characteristics of Unit Tests in Python:

- ***Isolation:*** Each test is isolated from others, testing only a single piece of functionality.
- ***Automation:*** Tests can be run automatically and repeatedly.
- ***Validation:*** Tests validate that the code behaves as expected under various conditions.
- Unit tests are a fundamental part of ensuring code quality and reliability. By testing individual units of code in isolation, developers can catch bugs early and ensure that each component behaves as expected. The "unittest" module provides a robust framework for writing and running these tests in Python.

  Example:



```python
import unittest

# Define the Calculator class
class Calculator:
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

    def multiply(self, a, b):
        return a * b

    def divide(self, a, b):
        if b == 0:
            raise ValueError("Cannot divide by zero!")
        return a / b
```



```python
# Define the test cases for the Calculator class
class TestCalculator(unittest.TestCase):

    def setUp(self):
        # Initialize a Calculator instance for testing
        self.calc = Calculator()

    def test_add(self):
        # Test addition method
        self.assertEqual(self.calc.add(2, 3), 5)
        self.assertEqual(self.calc.add(-1, 1), 0)
        self.assertEqual(self.calc.add(-1, -1), -2)

    def test_subtract(self):
        # Test subtraction method
        self.assertEqual(self.calc.subtract(10, 5), 5)
        self.assertEqual(self.calc.subtract(-1, 1), -2)
        self.assertEqual(self.calc.subtract(-1, -1), 0)

    def test_multiply(self):
        # Test multiplication method
        self.assertEqual(self.calc.multiply(3, 7), 21)
        self.assertEqual(self.calc.multiply(-1, 1), -1)
        self.assertEqual(self.calc.multiply(-1, -1), 1)

    def test_divide(self):
        # Test division method
        self.assertEqual(self.calc.divide(10, 2), 5)
        self.assertEqual(self.calc.divide(-10, 2), -5)
        self.assertEqual(self.calc.divide(-10, -2), 5)

        # Test division by zero
        with self.assertRaises(ValueError):
            self.calc.divide(10, 0)

# Run the tests
if __name__ == '__main__':
    unittest.main()
```

This code will run the tests and report any errors or failures in the implementation of the “Calculator” methods.

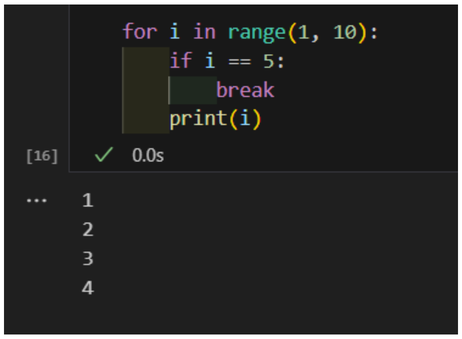
- What is break, continue and pass in Python?

  Answer:

  In Python, the “break”, “continue” and “pass” are control flow statements that are used to change the normal flow of loops and other control structures. These control flow statements help manage loop execution more precisely and are essential for writing efficient and readable code in Python. However, each of them serves a different purpose, which has been described in detail as follows:-

  - ***“break”:-*** The “break” statement is used to terminate and exit a loop prematurely, thereby transferring the control to the statement immediately following the loop.

    Example:

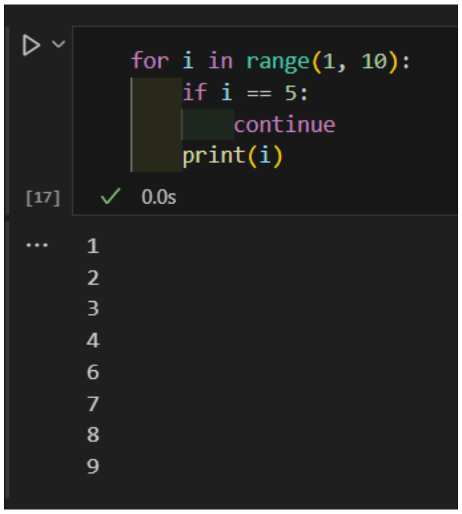


```
for i in range(1, 10):
    if i == 5:
        break
    print(i)
[16]  ✓ 0.0s
...  1
     2
     3
     4
```

From the given example, it is evident that as soon as the value of ‘i’ becomes 5, the loop stops executing and the ‘print’ statement after the break statement is not executed, when ‘i = 5’ and for any other subsequent value of ‘i’.

  - ***“continue”:-*** This statement is used to keep the rest of the code, inside a loop for the current iteration and proceed directly to the next iteration.

    Example:

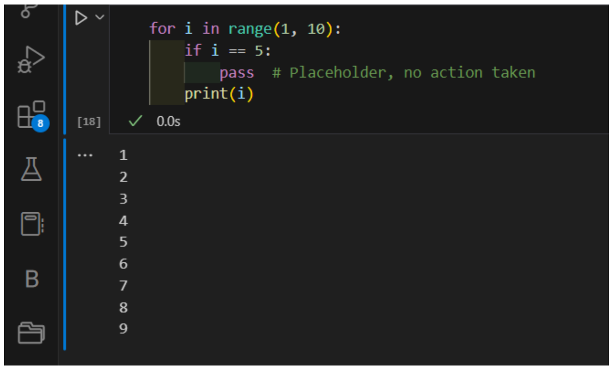


```
for i in range(1, 10):
    if i == 5:
        continue
    print(i)
[17]  ✓ 0.0s
...  1
     2
     3
     4
     6
     7
     8
     9
```

From the above example, it is evident, that when ‘i’ becomes 5, the continue statement is encountered and the ‘print’ statement is not executed and the loop moves to the next iteration.

- ***"pass":-*** This statement is a no-operation (NOP) statement. It is used as a placeholder in situations where a statement is syntactically required but no action is needed. This can be useful in scenarios like creating minimal class definitions, empty function bodies, or stubs for future code.

  Example:



```
for i in range(1, 10):
    if i == 5:
        pass  # Placeholder, no action taken
    print(i)
```

```
[18]  ✓ 0.0s
```

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

From the above example, it is evident that "pass" does nothing and the loop is executed in a normal manner.

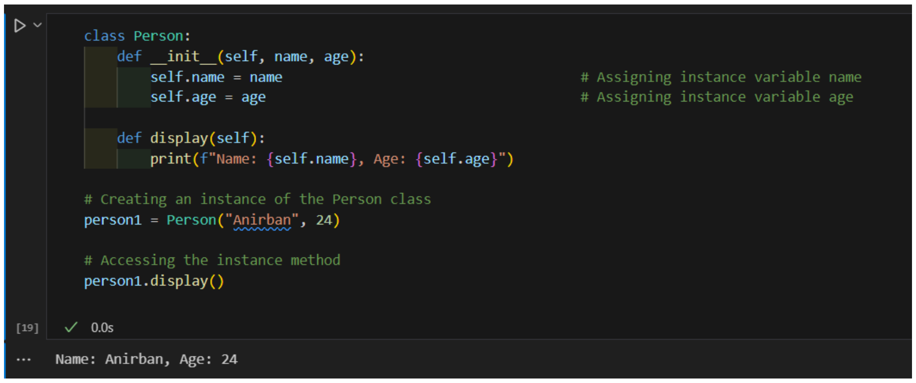
- What is the use of "self" in Python?

  Answer:

  The standard term for the first parameter of instance methods of a class in Python is "self". It is used to access variables and methods related to the instance and represents the class instance itself. The instance can refer to its own properties and functions by using "self".

  - "self" is a reference to the current instance of the class.
  - It is used to access instance variables and methods from within the class.
  - It is a convention to use "self" and it is the first parameter of instance methods.
  - "self" enables each instance of a class to maintain its own state and behavior.

  Example:



```
class Person:
    def __init__(self, name, age):
        self.name = name                                  # Assigning instance variable name
        self.age = age                                    # Assigning instance variable age

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

# Creating an instance of the Person class
person1 = Person("Anirban", 24)

# Accessing the instance method
person1.display()
```

```
[19]  ✓ 0.0s
```

```
Name: Anirban, Age: 24
```

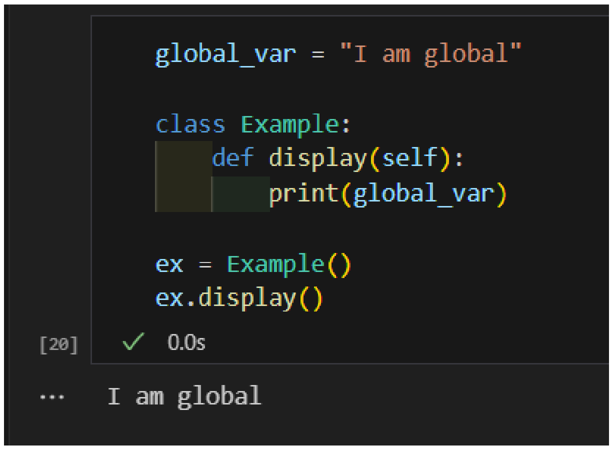
- What are global, protected and private attributes in Python?

  Answer:

  The explanation of Global, Protected and Private Attributes in Python, along with their examples are as follows:-

  ➢ ***Global Attributes:*** The term "global attributes" is not commonly used in relation to class attributes in Python, rather if an attribute is defined outside of a function or class, it may be a global variable. One can access these global variables from anywhere in the code, including functions and classes.

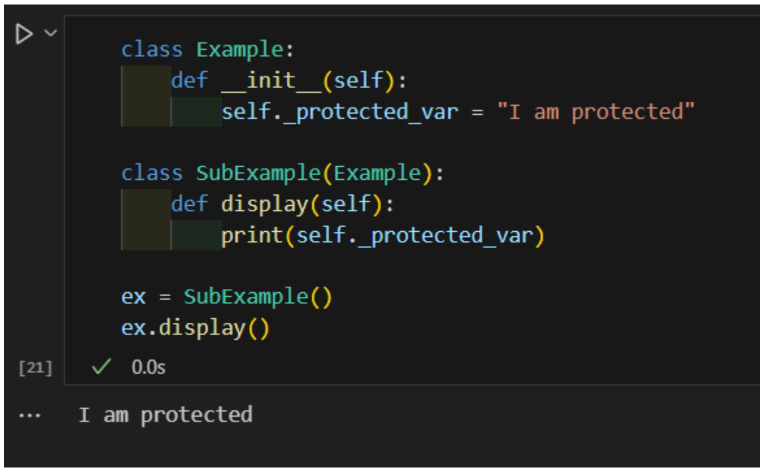
  Example:



```
global_var = "I am global"

class Example:
    def display(self):
        print(global_var)

ex = Example()
ex.display()
```
```
[20] ✓ 0.0s
... I am global
```

  ➢ ***Protected Attributes:*** Protected Attributes are intended for members of the class and its subclasses to have access to protected attributes. Protected attributes are conventionally prefixed with a single underscore ("_"). However, this is only a convention; access control is not enforced by it.

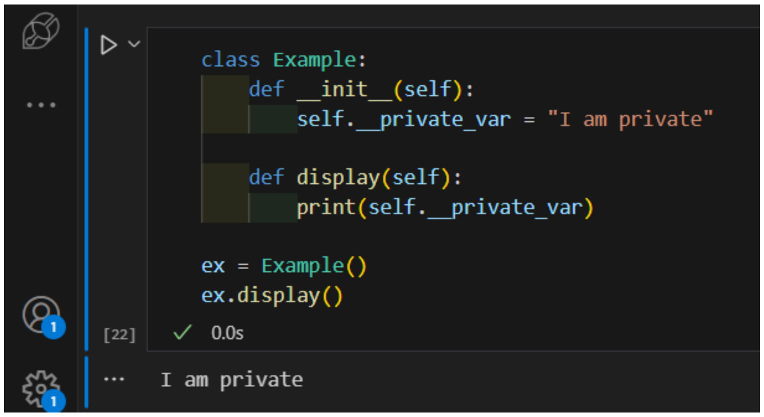
  Example:



```
class Example:
    def __init__(self):
        self._protected_var = "I am protected"

class SubExample(Example):
    def display(self):
        print(self._protected_var)

ex = SubExample()
ex.display()
```
```
[21] ✓ 0.0s
... I am protected
```

  ➢ ***Private Attributes:*** The purpose of private attributes is to restrict access to the class in which they are specified. They are made by adding a double underscore ("__") before the attribute name. As a result, the variable's name is changed to include the class name by the interpreter, causing name mangling and making it more difficult to access from outside the class.

Example:



```
class Example:
    def __init__(self):
        self.__private_var = "I am private"

    def display(self):
        print(self.__private_var)

ex = Example()
ex.display()
```

```
[22] ✓ 0.0s
```

```
I am private
```

While Python does not enforce strict access controls, following these conventions helps in writing clean, maintainable, and self-documenting code.

- What are modules and packages in Python?

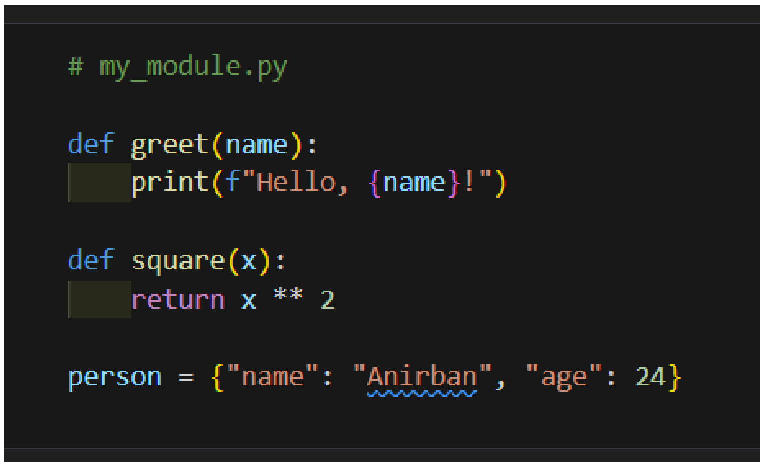
Answer:

In Python, Modules and Packages are organizational units that help structure, manage and organize Python code, avoiding any name conflicts and also promoting code reuse in Python projects.

The explanation of Modules and Packages, along with examples are as follows:-

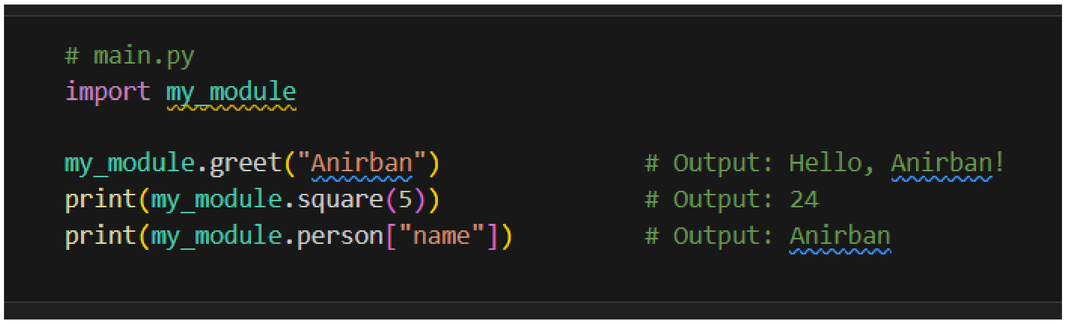
➢ ***Modules:*** A module is a single file containing Python code that defines classes, functions as well as variables, which allows to organize the Python code into reusable blocks that can be imported and used in other Python programs. Modules help in avoiding name clashes and provide a way to encapsulate related functionality.

Example:



```
# my_module.py

def greet(name):
    print(f"Hello, {name}!")

def square(x):
    return x ** 2

person = {"name": "Anirban", "age": 24}
```

Using a Module in Another Python File:



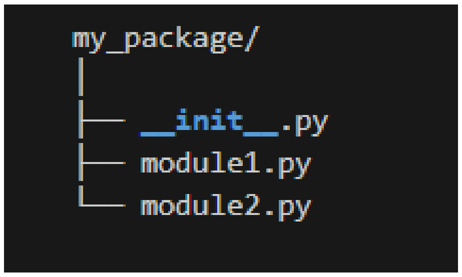
```
# main.py
import my_module

my_module.greet("Anirban")          # Output: Hello, Anirban!
print(my_module.square(5))          # Output: 24
print(my_module.person["name"])     # Output: Anirban
```

Here,

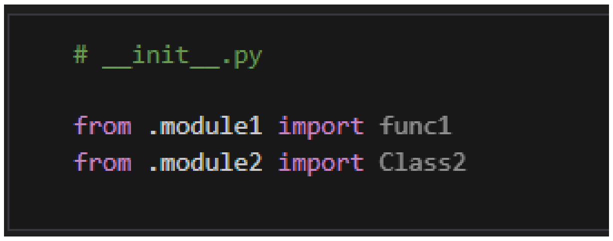
“my_module.py” is a module containing “greet”, “square” and “person”.
“main.py” imports “my_module” and accesses its functions and variables.

➢ ***Packages:*** Packages can be defined as directories containing multiple modules and an “__init__.py” file that might or might not be empty and if not empty, it can define the initialization code for the package.

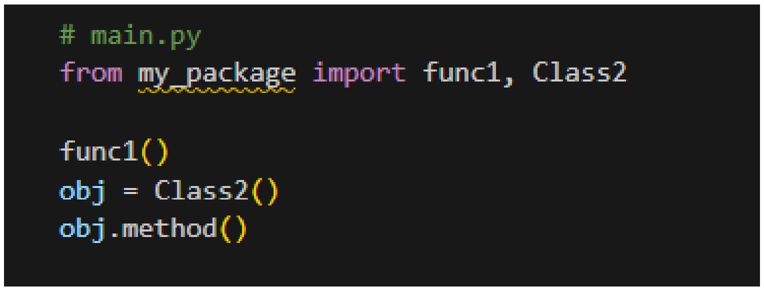
Example:



```
my_package/
│
├── __init__.py
├── module1.py
└── module2.py
```



```
# __init__.py

from .module1 import func1
from .module2 import Class2
```



```
# main.py
from my_package import func1, Class2

func1()
obj = Class2()
obj.method()
```

Here,

“my_package” is a package containing “module1.py” and “module2.py”
“__inti__.py” defines what is exposed when the package is imported.

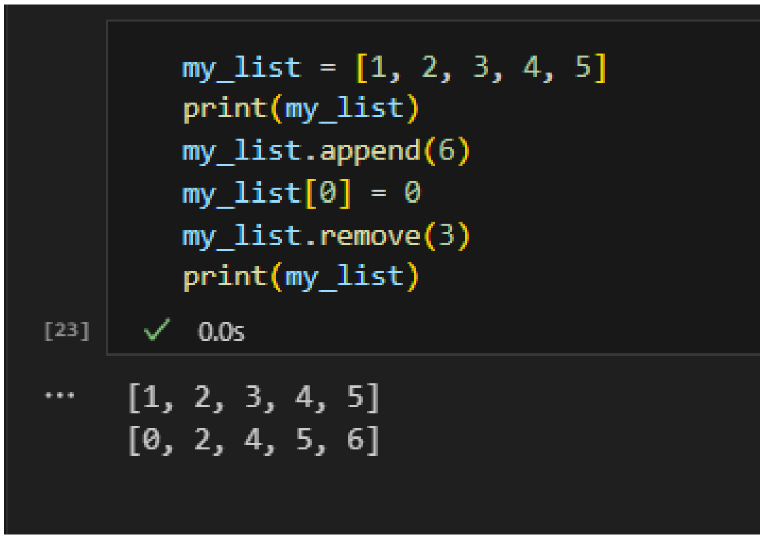
- What are lists and tuples? What is the key difference between the two?

  Answer:

  Both Lists and Tuples are Data Structures in Python, with varied properties and use cases, but are used to store a collection of items. Their explanations have been given as follows:-

  - ***Lists:*** A list is an ordered collection of items that are mutable, meaning the items can be changed, added, or removed.

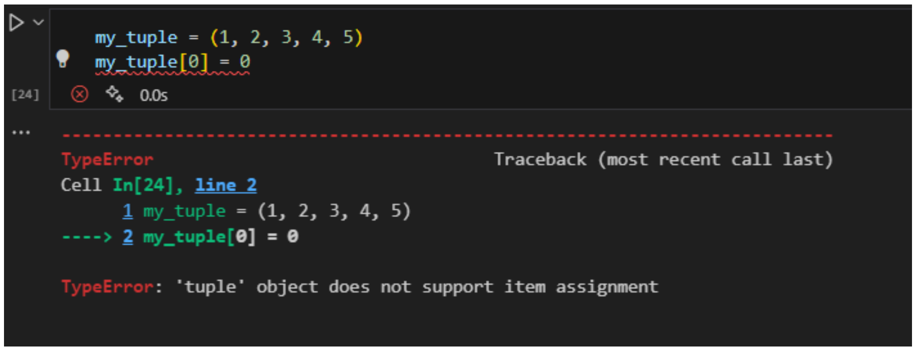
    Example:



```
my_list = [1, 2, 3, 4, 5]
print(my_list)
my_list.append(6)
my_list[0] = 0
my_list.remove(3)
print(my_list)
[23]  ✓  0.0s

...  [1, 2, 3, 4, 5]
     [0, 2, 4, 5, 6]
```

  - ***Tuples:*** A tuple is an ordered collection of immutable items, meaning once a tuple is created, its elements cannot be changed, added, or removed.

    Example:



```
my_tuple = (1, 2, 3, 4, 5)
my_tuple[0] = 0
[24]  0.0s

...
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[24], line 2
      1 my_tuple = (1, 2, 3, 4, 5)
----> 2 my_tuple[0] = 0

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

The key differences between Lists and Tuples are as follows:-

Features or Parameters	***Lists***	***Tuples***
Mutability	Mutable, i.e., elements can be modified, removed or added.	Immutable, i.e., elements cannot be modified, removed or added.
Syntax	Defined using square brackets “[]”	Defined using parenthesis “()”
Methods	Supports various methods like “append()”, “remove()”, “pop()”, “extend()”, “insert()”, etc..	Supports only a limited number of methods like “index()” and “count()”.
Size	Lists can grow or shrink in size as elements are added or removed, i.e., they are dynamic in nature.	Once a tuple is created, it cannot be resized.
Performance	Slightly slower than tuples due to their dynamic nature and mutability.	Generally faster due to their immutability and fixed size.
Use Cases	Preferred when a collection of items, subject to change over time, needs to be stored.	Preferred for collection of items that should not change, ensuring data integrity.

- What is an Interpreted language & dynamically typed language? Write 5 differences between them.

  Answer:

  The description of Interpreted Language and Dynamically Typed Language are as follows:-

  ➢ ***Interpreted Language:*** When a programming language is interpreted, the majority of its implementations carry out instructions freely and immediately without requiring them to first be compiled into a machine language program. An interpreter reads and runs the source code rather than compiling it into machine code.

  Characteristics of Interpreted Language:
  - ✓ *Execution:* Code is executed line-by-line by an interpreter.
  - ✓ *Portability:* More portable as the interpreter abstracts away machine-specific details.
  - ✓ *Development Speed:* Generally faster development cycles as there is no need for a separate compilation step.

- ✓ *Error Handling:* Errors are caught at runtime rather than at compile time.
- ✓ *Flexibility:* Easier to implement and modify on the fly.

- ➢ ***Dynamically Typed Language:*** Programming languages that do type checking at runtime instead of compile time are known as dynamically typed languages. This implies that when writing your program, you are not need to define the type of a variable. Rather, the type is linked to the value that the variable contains.

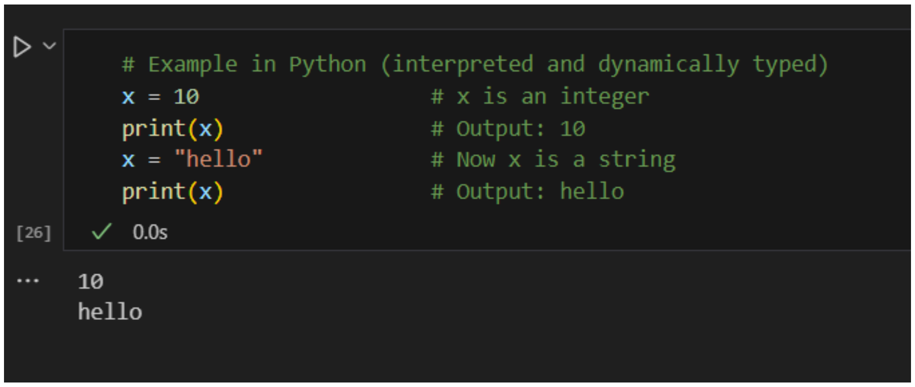
  Characteristics of Dynamically Typed Language:
  - ✓ *Type Checking:* Types are checked during runtime.
  - ✓ *Ease of Use:* No need to declare variable types explicitly.
  - ✓ *Scripting and Prototyping:* Particularly useful for scripting and rapid prototyping.
  - ✓ *Error Detection:* Type-related errors are detected only when the relevant code is executed.
  - ✓ *Flexibility:* Variables can hold any type of data at different times during execution.

Five key differences between Interpreted Language and Dynamically Typed Language are as follows:-

Features or Parameters	***Interpreted Language***	***Dynamically Typed Language***
Type Checking	Interpreted languages execute code directly without focusing on type-checking.	Dynamically typed languages perform type checking at runtime, not at compile time.
Compilation	Interpreted languages do not require compilation into machine code before execution.	Dynamically typed languages can be interpreted or compiled, but they determine variable types during execution.
Variable Declaration	Interpreted languages can be statically or dynamically typed and are not defined by variable declaration requirements.	Dynamically typed languages do not require explicit variable type declarations.
Error Detection	In interpreted languages, syntax errors are detected at runtime as the interpreter encounters them.	In dynamically typed languages, type errors are detected at runtime when the relevant code is executed.
Execution Speed	Interpreted languages generally have slower execution speeds due to the overhead of runtime interpretation.	Dynamically typed languages may have variable execution speeds, influenced by the need to determine types at runtime.

Example of Overlap:

Python is an overlap, i.e., it is both an interpreted language as well as a dynamically typed language, since the Python code is executed by the Python interpreter and while typing the code, variables are not declared explicitly and types are determined at runtime.



```
# Example in Python (interpreted and dynamically typed)
x = 10                    # x is an integer
print(x)                  # Output: 10
x = "hello"               # Now x is a string
print(x)                  # Output: hello
```

```
[26]  ✓ 0.0s
10
hello
```

- What are Dict and List comprehensions?

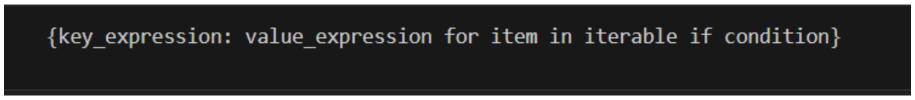
  Answer:

  A comparative study between Dict Comprehension and List Comprehension are as follows:-

Features or Parameters	***Dict Comprehension***	***List Comprehension***
Definition	By applying an expression to each item in an existing iterable and optionally filtering items with a conditional statement, dict comprehensions offer a condensed method for creating dictionaries in Python.	In Python, list comprehensions offer a condensed method for creating lists. By applying an expression to each item in an existing iterable and optionally filtering items with a conditional statement, a new list can be created.
Representation	Use curly braces "{}" with a "key: value" pair.	Use square brackets "[]".
Use Cases	Ideal for creating and transforming dictionaries.	Ideal for creating and transforming lists.
Conciseness	Offer a more concise way to create dictionaries.	Provide a more compact and readable way to create lists compared to traditional loops.

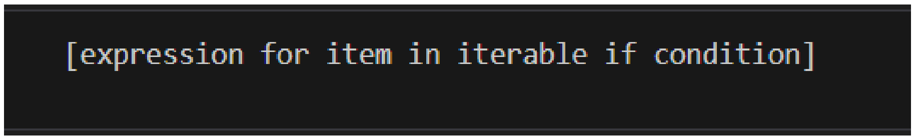
Syntax:

Dict Comprehension:



```
{key_expression: value_expression for item in iterable if condition}
```

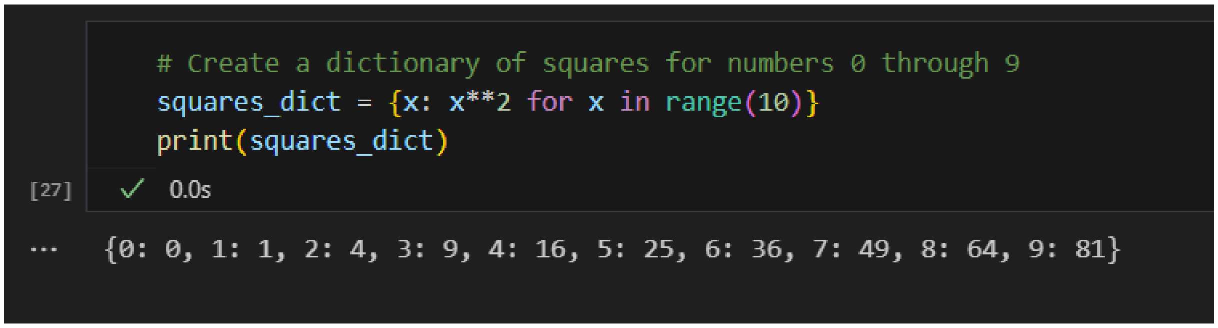
List Comprehension:



```
[expression for item in iterable if condition]
```

Examples:

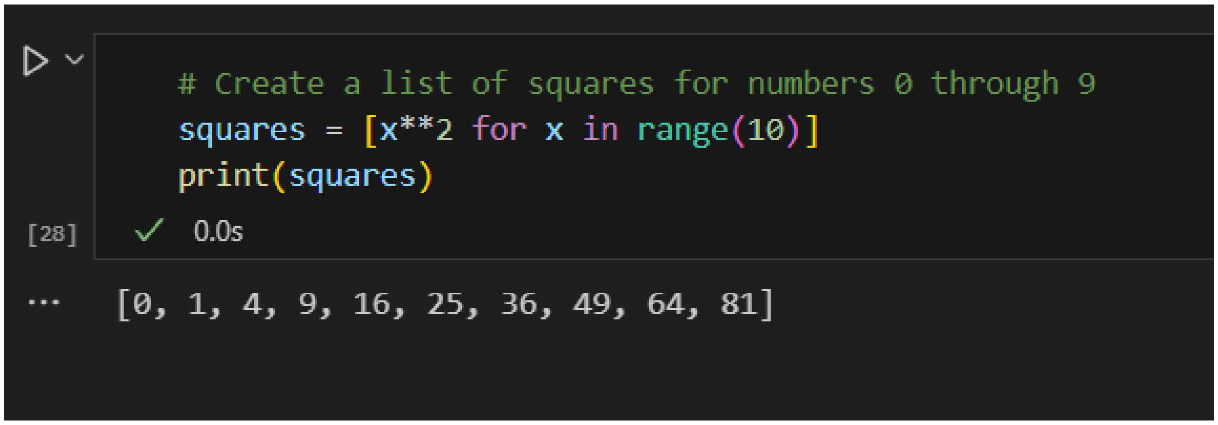
Dict Comprehension:



```
# Create a dictionary of squares for numbers 0 through 9
squares_dict = {x: x**2 for x in range(10)}
print(squares_dict)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

List Comprehension:



```
# Create a list of squares for numbers 0 through 9
squares = [x**2 for x in range(10)]
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

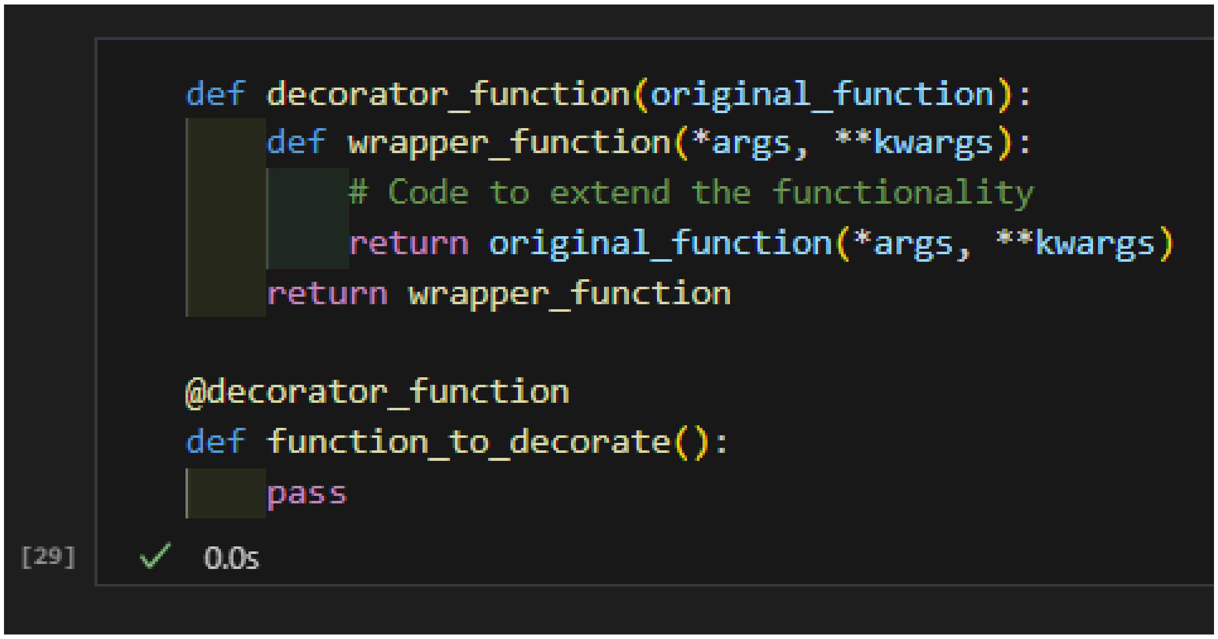
- What are decorators in Python? Explain it with an example. Write down its use cases.

  Answer:

  Python decorators are a type of design pattern that allows to change methods or functions by utilizing other functions. They offer an adaptable means of increasing a function or method's functionality without having to alter it permanently. It is common practice to add features like caching, timing, authentication, and logging using decorators. Python decorators offer a strong and sophisticated means of enhancing or changing the behaviour of functions or methods. Modularity, cleanliness can be maintained and the code can be reused by utilizing decorators.

  Syntax: A decorator is applied to a function by prefixing the function definition with the “@decorator_name” syntax.

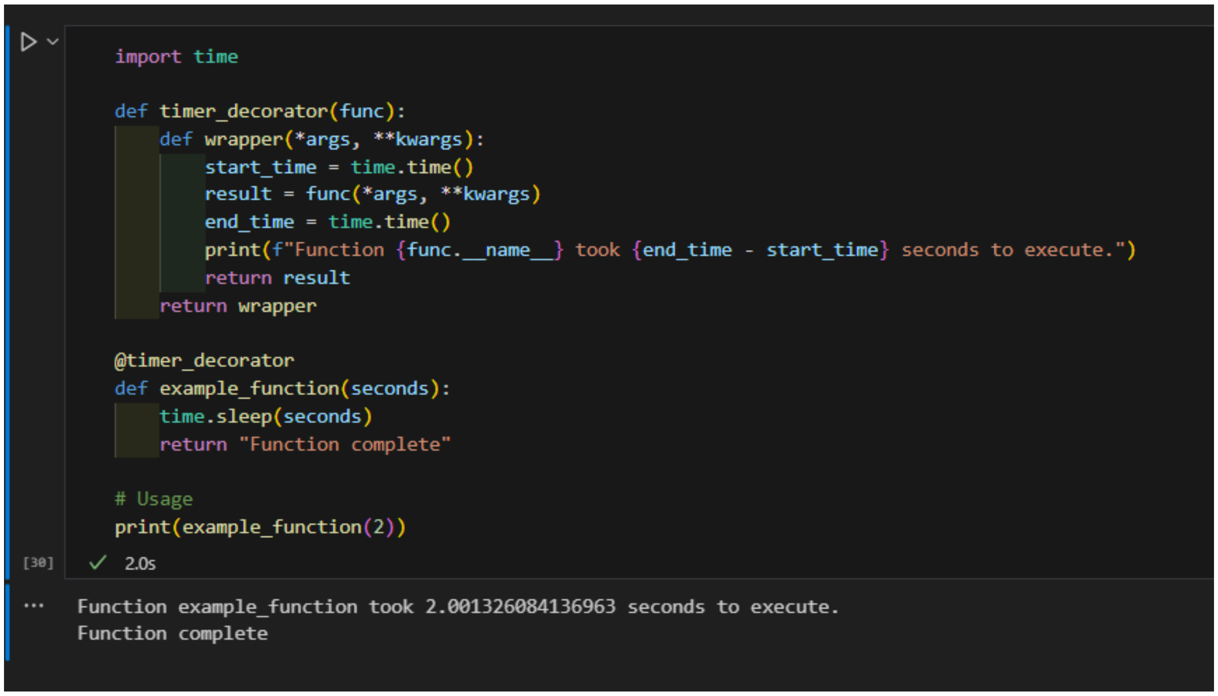
  Basic Structure:



```
def decorator_function(original_function):
    def wrapper_function(*args, **kwargs):
        # Code to extend the functionality
        return original_function(*args, **kwargs)
    return wrapper_function

@decorator_function
def function_to_decorate():
    pass
```

```
[29]  ✓ 0.0s
```

Example: A simple decorator that logs the execution time of a function.



```python
import time

def timer_decorator(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        print(f"Function {func.__name__} took {end_time - start_time} seconds to execute.")
        return result
    return wrapper

@timer_decorator
def example_function(seconds):
    time.sleep(seconds)
    return "Function complete"

# Usage
print(example_function(2))
```

[30] ✓ 2.0s

```
Function example_function took 2.001326084136963 seconds to execute.
Function complete
```

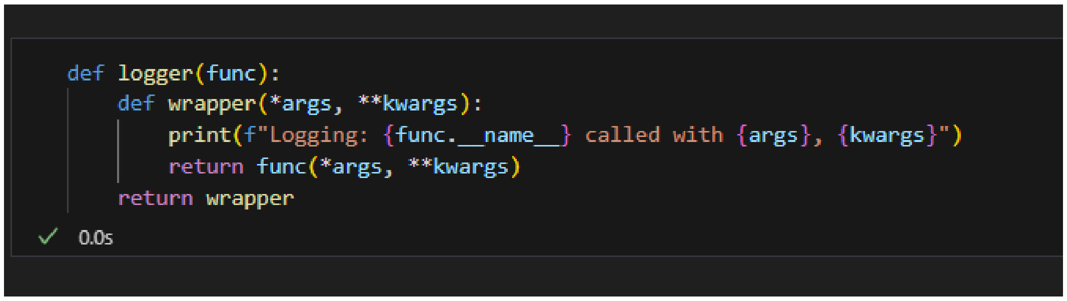
Explanation:

- ***"timer_decorator":*** This is our decorator function. It takes another function "func" as an argument and defines a "wrapper" function inside it.
- ***wrapper:*** This function adds the additional functionality (timing the execution) before and after calling the original function "func".
- ***"@timer_decorator":*** This syntax applies the decorator to "example_function", effectively replacing "example_function" with "wrapper".
- When "example_function(2)" is called, it will print the time taken to execute the function.

Use Cases: The use cases are as follows:-

- ***Logging:*** Decorators can be used to log the entry, exit, and parameters of a function for debugging and monitoring purposes.

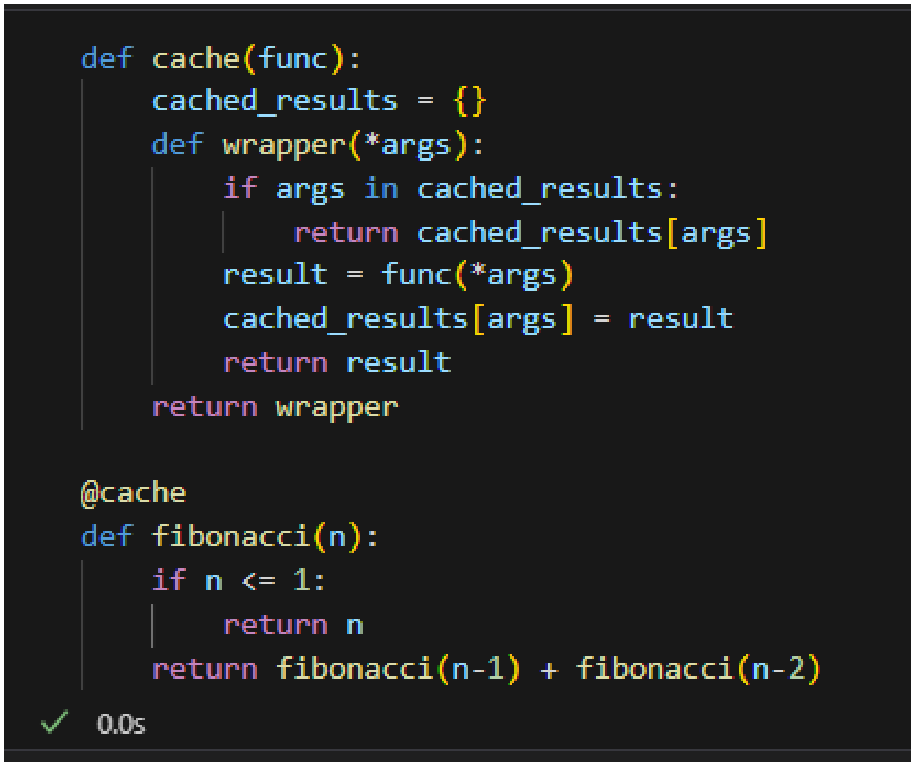
  Code Snippet:



```python
def logger(func):
    def wrapper(*args, **kwargs):
        print(f"Logging: {func.__name__} called with {args}, {kwargs}")
        return func(*args, **kwargs)
    return wrapper
```

✓ 0.0s

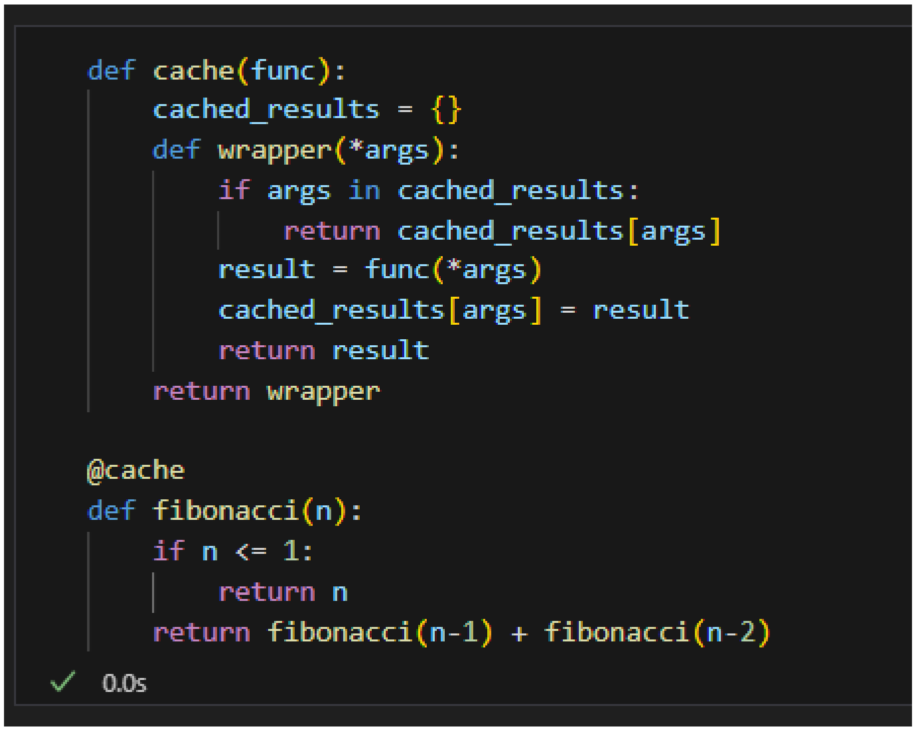
- ***Authorization:*** Decorators can enforce user permissions or roles before allowing a function to execute.
  Code Snippet:



```python
def cache(func):
    cached_results = {}
    def wrapper(*args):
        if args in cached_results:
            return cached_results[args]
        result = func(*args)
        cached_results[args] = result
        return result
    return wrapper

@cache
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)
```

```
✓ 0.0s
```

- ***Caching:*** Decorators can be used to cache the results of expensive function calls to improve performance.
  Code Snippet:

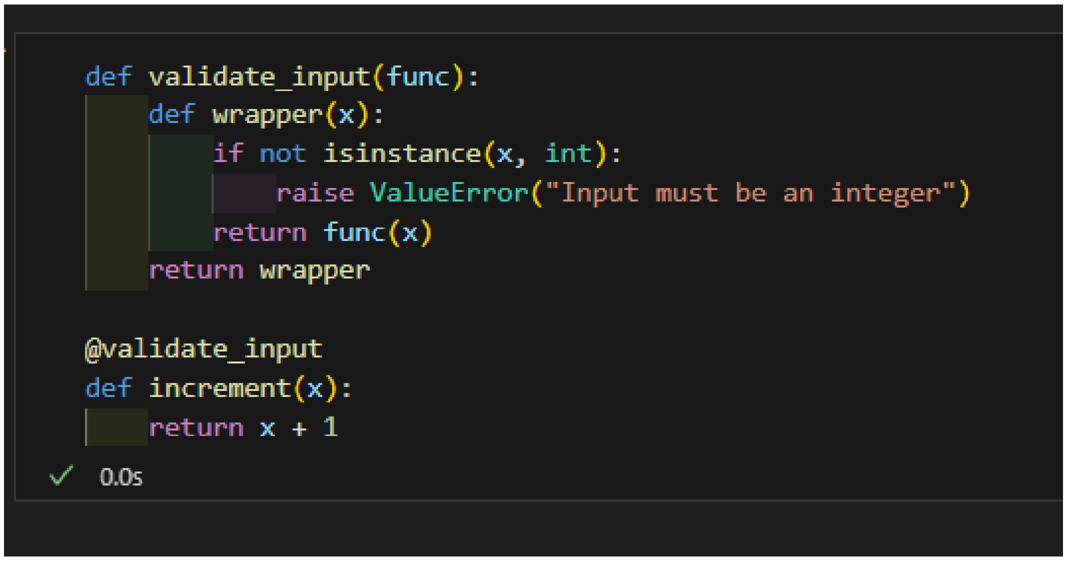


```python
def cache(func):
    cached_results = {}
    def wrapper(*args):
        if args in cached_results:
            return cached_results[args]
        result = func(*args)
        cached_results[args] = result
        return result
    return wrapper

@cache
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)
```

```
✓ 0.0s
```

- ***Validation:*** Decorators can validate inputs to a function before the function executes.

  Code Snippet:



```
def validate_input(func):
    def wrapper(x):
        if not isinstance(x, int):
            raise ValueError("Input must be an integer")
        return func(x)
    return wrapper

@validate_input
def increment(x):
    return x + 1
✓ 0.0s
```

- ***Timing:*** Decorators can measure and report the execution time of functions, as shown in the earlier example.

- How is memory managed in Python?

  Answer:

  The memory management system of Python is designed to be transparent and efficient to the user, allowing them to focus on writing code, without worrying about the details of low-level memory management, although they can interact with the garbage collector and inspect memory usage using Python's built-in modules. Python handles memory management through a combination of reference counting and garbage collection. The Python memory manager automatically allocates and deallocates memory as needed. This automatic memory management is one of the reasons why Python is considered a high-level, easy-to-use language.

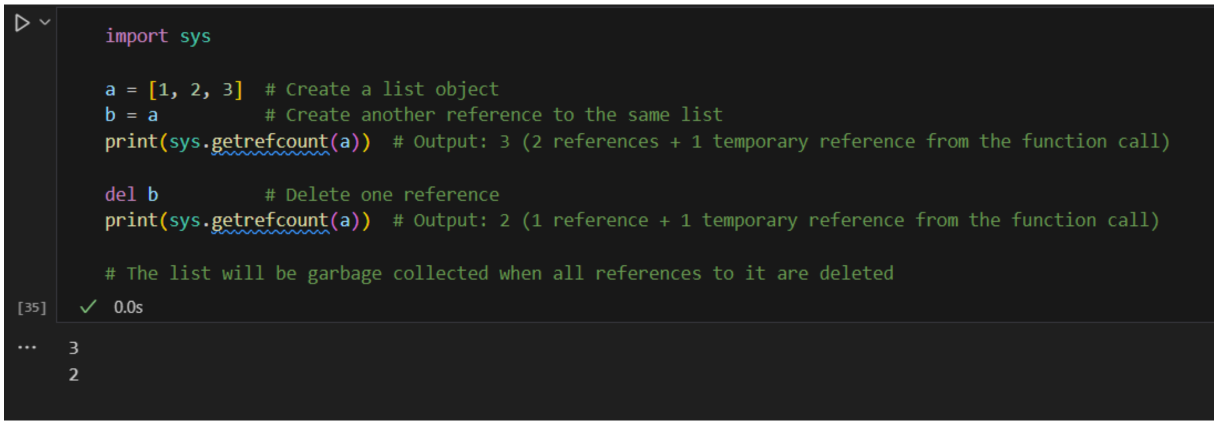
  Memory management in Python works in the following ways:-

  - ***Python Memory Manager:***
    - ✓ *Automatic Memory Location:* When a new object is created in Python, the Python memory manager automatically allocates the necessary memory.
    - ✓ *Object Lifecycle Management:* The memory manager tracks all objects in the program and their references, managing the lifecycle of each object.
  - ***Reference Counting:***

- ✓ *Reference Count:* Python uses a reference counting mechanism to track the number of references to each object. Every object has an associated reference count.
- ✓ *Incrementing the Reference Count:* The reference count of an object increases when it is assigned to a new variable or passed as an argument to a function.
- ✓ *Decrementing the Reference Count:* The reference count decreases when a reference to the object is deleted or goes out of scope.
- ✓ *Garbage Collection:* When an object's reference count drops to zero, meaning it is no longer accessible, Python automatically deallocates the memory for that object.

- ➢ ***Garbage Collection:***
  - ✓ *Cycle Detection:* While reference counting is efficient, it cannot handle reference cycles (where two or more objects reference each other). To address this, Python includes a cyclic garbage collector that identifies and collects these cycles.
  - ✓ *Generational Garbage Collection:* Python's garbage collector divides objects into three generations based on their lifespan – Generation 0 (just created); Generation 1 (contains older objects that have survived one round of garbage collection) and Generation 2 (oldest objects that have survived multiple rounds of garbage collection).
  - ✓ *Optimization:* The garbage collector prioritizes collecting objects in the younger generations more frequently since most objects have a short lifespan.
- ➢ ***Memory Pools and Arenas:***
  - ✓ *Memory Pools:* Python organizes memory into pools, where each pool can hold objects of a specific size range. This helps reduce memory fragmentation and improves allocation efficiency.
  - ✓ *Arenas:* Pools are further grouped into larger blocks of memory called arenas. Arenas are the units of memory that the operating system manages directly.
- ➢ ***Python's Memory Management Modules:***
  - ✓ *"gc" Module:* Python provides the "gc" module, which allows developers to interact with the garbage collector. One can manually trigger garbage collection, inspect objects, and adjust the garbage collector's behavior.
  - ✓ *"sys" Module:* The "sys" module provides functions like "sys.getrefcount()" to inspect the reference count of an object, and "sys.getsizeof()" to get the memory size of an object.

Example:



```
import sys

a = [1, 2, 3]  # Create a list object
b = a          # Create another reference to the same list
print(sys.getrefcount(a))  # Output: 3 (2 references + 1 temporary reference from the function call)

del b          # Delete one reference
print(sys.getrefcount(a))  # Output: 2 (1 reference + 1 temporary reference from the function call)

# The list will be garbage collected when all references to it are deleted
```

```
[35] ✓ 0.0s
3
2
```

Practical Memory Management Tips

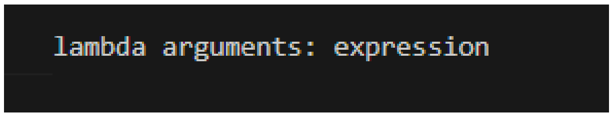
- ***Avoiding Memory Leaks:*** Ensure objects are dereferenced when no longer needed to prevent memory leaks, especially in long-running applications.
- ***Use of Generators:*** Generators are memory efficient as they generate values on-the-fly and do not store the entire sequence in memory.
- ***Explicit Deallocation:*** Use "del" to explicitly delete objects and free up memory when their usage is complete.
- ***Memory Profiling and Optimization:*** Use tools like "memory_profiler" and "tracemalloc" to profile and optimize memory usage in your Python applications, especially for large-scale or memory-intensive tasks.

- What is lambda in Python? Why is it used?

Answer:

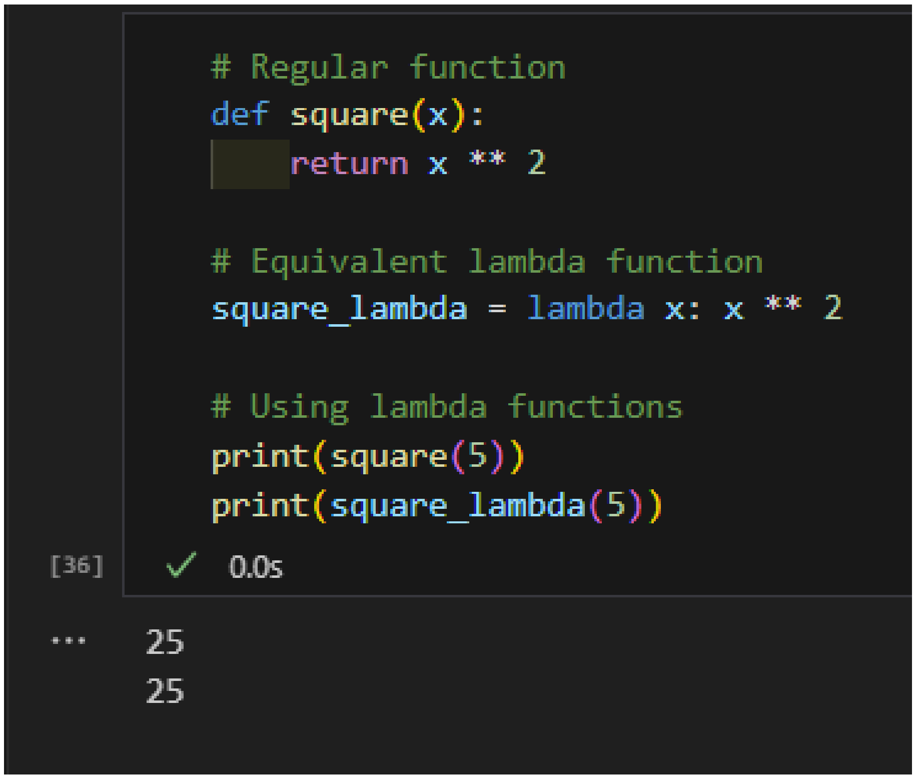
Lambda functions in Python are small, anonymous functions, defined using the "lambda" keyword. These functions provide a way to create small, anonymous functions inline, without explicitly naming them using "def". They are used for simple tasks where defining a named function would be overkill or less readable. Their compact syntax and ability to be used immediately where needed make them particularly useful in certain contexts, especially in functional programming and when working with higher-order functions.

Syntax:



```
lambda arguments: expression
```

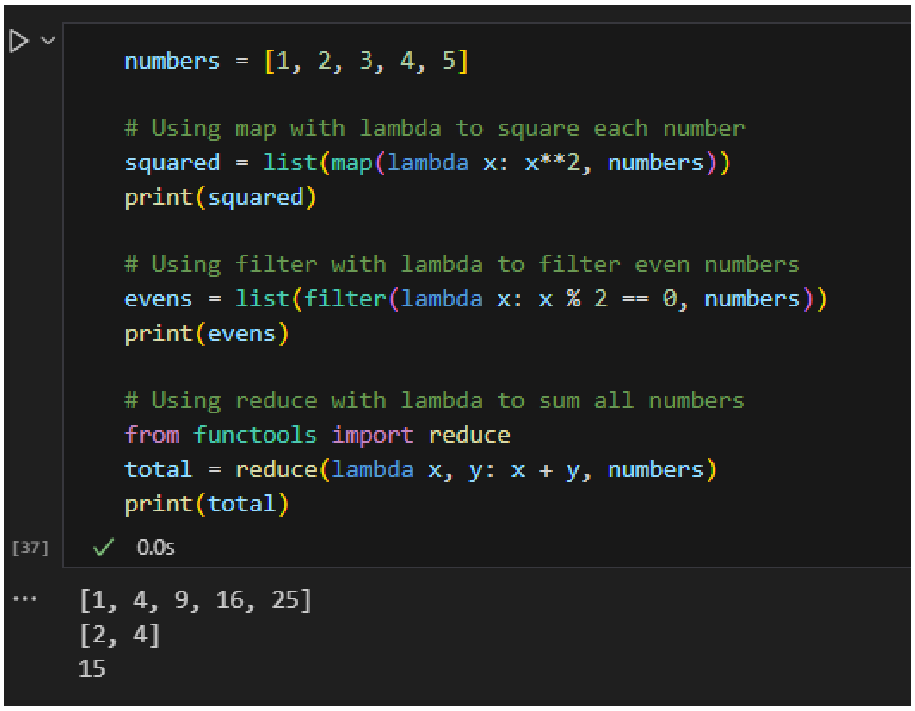
Example:
Comparison between Regular Function and Lambda Function



```
# Regular function
def square(x):
    return x ** 2

# Equivalent lambda function
square_lambda = lambda x: x ** 2

# Using lambda functions
print(square(5))
print(square_lambda(5))
```

```
[36] ✓ 0.0s

25
25
```

Practical Use Case
Lambda functions can be used with higher-order functions like “map()”, “filter()” and “reduce()”.



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

# Using map with lambda to square each number
squared = list(map(lambda x: x**2, numbers))
print(squared)

# Using filter with lambda to filter even numbers
evens = list(filter(lambda x: x % 2 == 0, numbers))
print(evens)

# Using reduce with lambda to sum all numbers
from functools import reduce
total = reduce(lambda x, y: x + y, numbers)
print(total)
```

```
[37] ✓ 0.0s

[1, 4, 9, 16, 25]
[2, 4]
15
```

Lambda Functions are used for the following reasons:-

- ***Anonymous Function:*** Lambda functions are anonymous, meaning they don't require a name like regular functions defined with "def".
- ***Conciseness:*** They allow to write functions in a more compact way, especially for simple operations.
- ***Functional Programming:*** Lambda functions are commonly used in functional programming paradigms where functions are treated as first-class citizens.
- ***Immediate Use:*** They are useful when it is needed to quickly define and use a simple function without defining a separate named function.

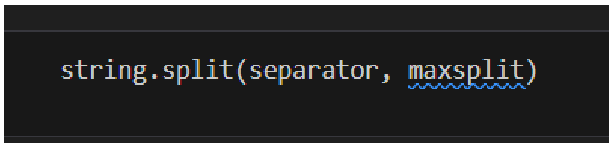
- Explain "split()" and "join()" functions in Python.

Answer:

The explanation of "split()" and "join()" functions in Python are as follows:-

- ***"split()":*** The "split()" function in Python is used to break a string into a list of substrings based on a specified delimiter. If no delimiter is specified, it defaults to splitting by any whitespace (spaces, tabs, newlines, etc.).

Syntax:



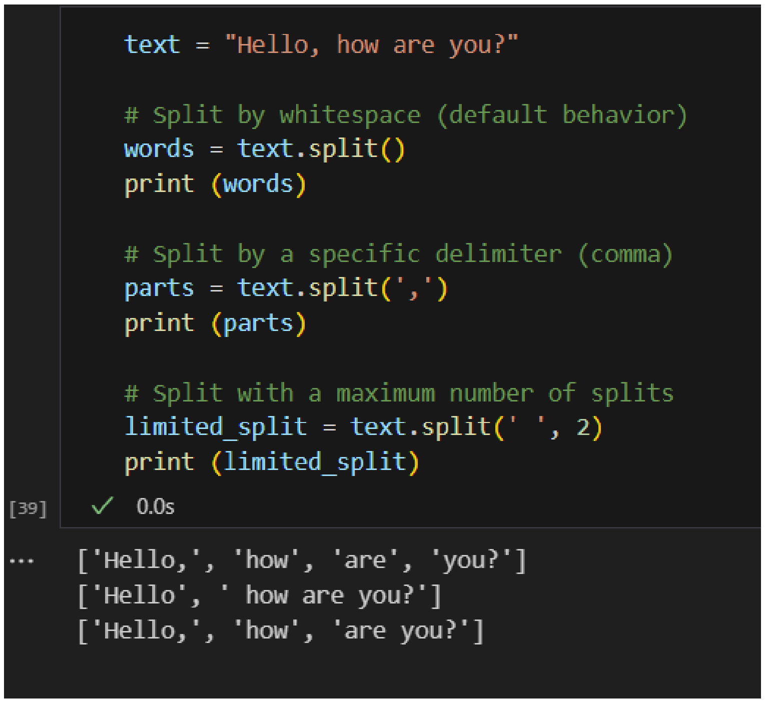
```
string.split(separator, maxsplit)
```

Here,

'separator' → it is the delimiter on which the string needs to be split, but if it is not provided, then whitespaces are considered to be the delimiters, by default, i.e., it is optional.

'maxsplit' → it specifies the maximum number of splits and the list will have at most 'maxsplit + 1' elements and it is also optional.

Example:



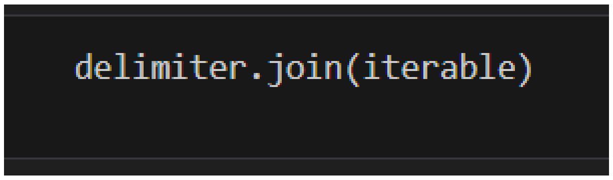
```
text = "Hello, how are you?"

# Split by whitespace (default behavior)
words = text.split()
print (words)

# Split by a specific delimiter (comma)
parts = text.split(',')
print (parts)

# Split with a maximum number of splits
limited_split = text.split(' ', 2)
print (limited_split)
```

```
[39]  ✓ 0.0s
```

```
['Hello,', 'how', 'are', 'you?']
['Hello', ' how are you?']
['Hello,', 'how', 'are you?']
```

➢ ***"join()":*** The "join()" function in Python is used to concatenate a list (or any iterable) of strings into a single string, with a specified delimiter between each element.

Syntax:



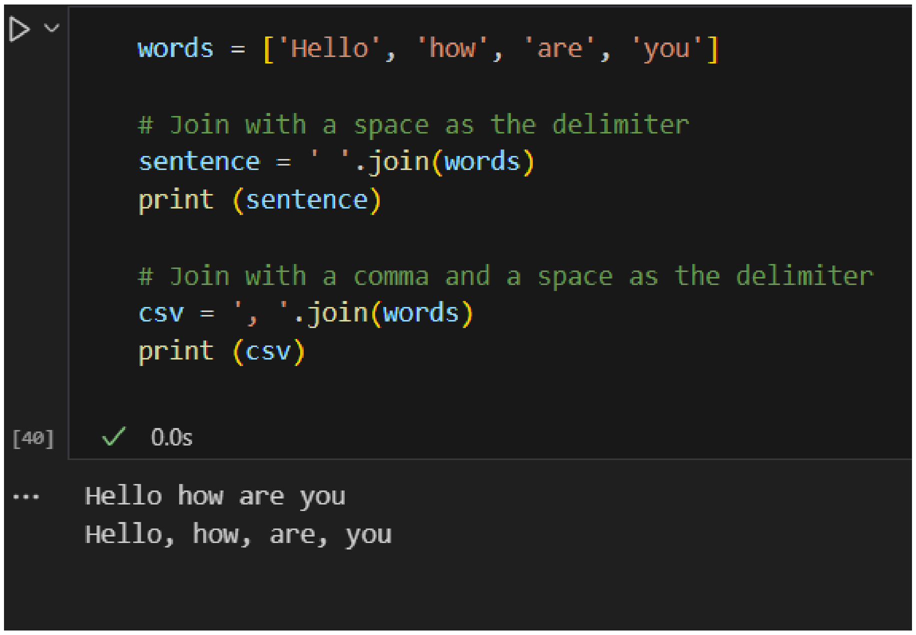
```
delimiter.join(iterable)
```

Here.

'delimiter' → refers to the string to be inserted between each element of the iterable

'iterable' → refers to any iterable object like list, tuple, strings.

Example:



```
words = ['Hello', 'how', 'are', 'you']

# Join with a space as the delimiter
sentence = ' '.join(words)
print (sentence)

# Join with a comma and a space as the delimiter
csv = ', '.join(words)
print (csv)
```

[40] ✓ 0.0s

```
Hello how are you
Hello, how, are, you
```

- What are iterators , iterable & generators in Python?

  Answer:

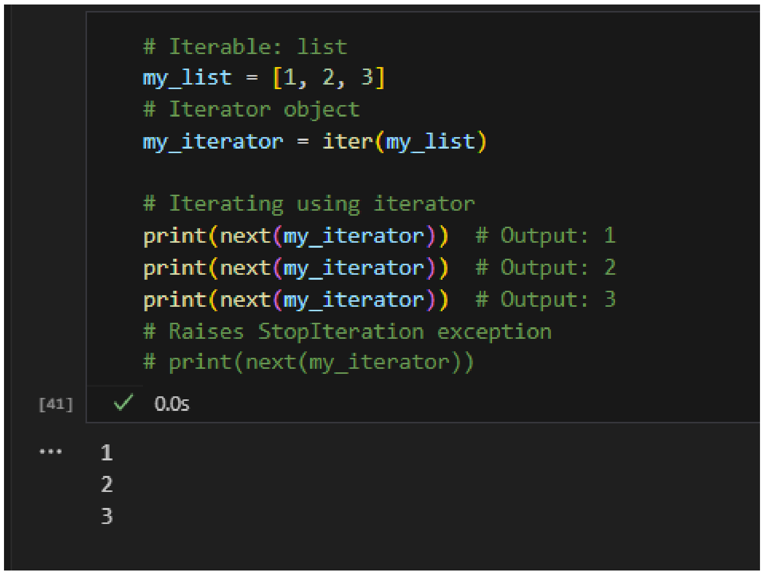
  Iterators in Python: An iterator is an object with a countable number of items that can be traversed through, allowing to go through each value in turn. An object that implements the iterator protocol in Python and has the methods "__iter__()" and "__next__()" is called an iterator. Iterators are useful when it is required to define a custom iteration behaviour.

  Iterable in Python: A Python object that can be iterated over in a for-loop is called an iterable if it can return each of its members one at a time. Typically, dictionaries, strings, tuples, and lists are examples of iterables. If an object supports the "__iter__()" function, then it is regarded as iterable. All it takes is an object that has the ability to return an iterator to define an iterable.It produces an iterator object upon implementation of the "__iter__()" method. Iterables are commonly used for data structures that are a collection of items, like lists, tuples, dictionaries and sets.

  Generators in Python: Generators are an easy-to-use yet effective technique for making iterators. Although they are defined with standard function syntax, they always return data by utilising the "yield" statement. The generator will return the subsequent value upon each call to its "__next__()" method, pausing execution at the "yield" statement until the generator receives a new value request. Because they only manufacture things one at a time and when needed, generators consume less memory. Generators are ideal for handling large datasets or streams of data where loading everything into memory at once isn't feasible and they are also useful for defining simple iterators without creating a full iterator class.

Example:

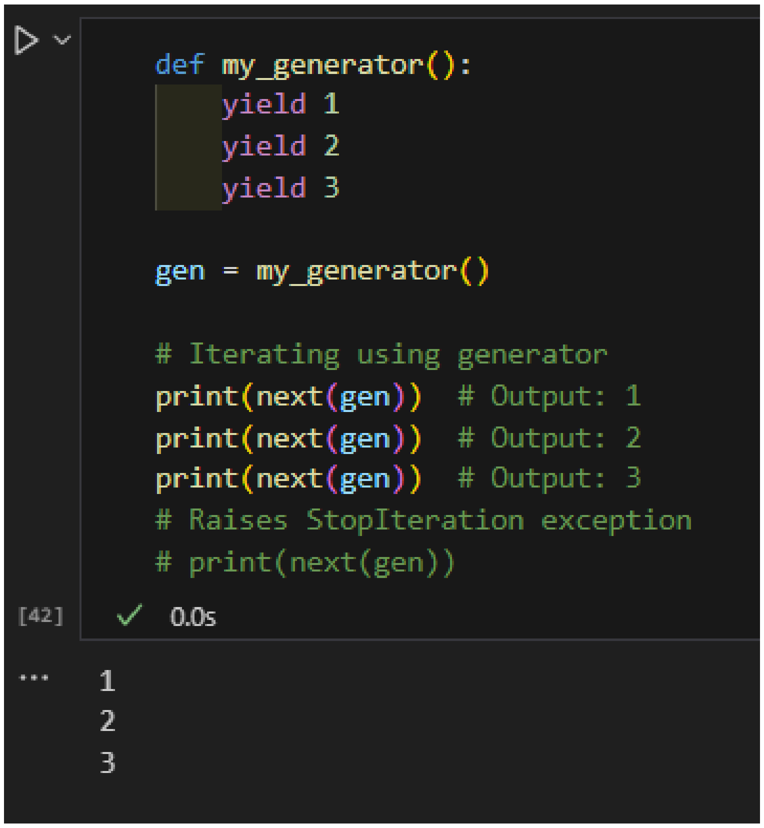
Iterable and Iterator:



```
# Iterable: list
my_list = [1, 2, 3]
# Iterator object
my_iterator = iter(my_list)

# Iterating using iterator
print(next(my_iterator))  # Output: 1
print(next(my_iterator))  # Output: 2
print(next(my_iterator))  # Output: 3
# Raises StopIteration exception
# print(next(my_iterator))
```

```
[41] ✓ 0.0s
1
2
3
```

Generator:



```
def my_generator():
    yield 1
    yield 2
    yield 3

gen = my_generator()

# Iterating using generator
print(next(gen))  # Output: 1
print(next(gen))  # Output: 2
print(next(gen))  # Output: 3
# Raises StopIteration exception
# print(next(gen))
```

```
[42] ✓ 0.0s
1
2
3
```

- What is the difference between xrange and range in Python?
  Answer:

The difference between “xrange” and “range” in Python, primarily comes down to the version of Python being used, so the distinction is given below:-

Version of Python	***xrange***	***range***
Python 2	“xrange” returns an “xrange” object, which generates the numbers on-the-fly as iteration takes place, rather than creating the entire list in memory at once and this makes “xrange” more memory-efficient for large ranges.	“range: generates a list containing all the numbers specified in the range. For example, “range(1, 5)” produces “[1, 2, 3, 4]”, which means that range creates a full list in memory, which can be inefficient for large ranges.
Python 3	The “xrange” function has been removed, as its functionality has been integrated into the range function.	In Python 3, “range” behaves like “xrange” of Python 2 as it returns a range object that generates numbers on-the-fly (lazily) as you iterate over it, without creating a full list in memory.

- Pillars of Oops.
  Answer:
  Object-Oriented Programming (OOP) is a programming paradigm that uses objects and classes to structure software programs. It is built on four main principles, often referred to as the pillars of OOP. These principles collectively facilitate the development of robust, maintainable, and scalable software by promoting clear structure, code reuse, and flexibility.

These pillars are presented in a tabular format as follows:-

Features or Parameters	***Encapsulation***	***Abstraction***	***Inheritance***	***Polymorphism***
Definition	Encapsulation is the process of combining methods (functions) that manipulate data with data (attributes) into a single unit called a class. As a precaution against unintentional interference and misuse of the techniques and data, it also entails limiting direct access to parts of an object's components.	The idea of abstraction is to present an object's essential qualities alone, while concealing its intricate implementation details. This eliminates the requirement for the user to comprehend the underlying complexity and enables higher level interaction with the object.	The process of one class (the child or subclass) inheriting properties and methods from another class (the parent or superclass) is known as inheritance. As a result, there is a natural hierarchy between classes and code reuse is encouraged.	The capacity of various classes to be viewed as instances of the same class via a shared interface is known as polymorphism. It permits the use of a single interface for a broad class of activities, with the particular action being determined by the particular circumstances
Purpose	To protect the internal state of an object and to promote modularity and maintainability.	To reduce complexity and allow the programmer to focus on interactions at a higher level.	To facilitate code reuse and create a relationship between classes.	To enable a single interface to represent different underlying forms (data types).
Example	In a class, one may use private attributes and public methods to provide access to those attributes in a	A car object might abstract away the details of its engine and expose methods like “start()”,	A “Vehicle” class could be a parent class with “Car” and “Bike” as subclasses, inheriting	A “shape” interface might have methods like “draw()”. Different classes like “Circle”, “Square”, and “Triangle”

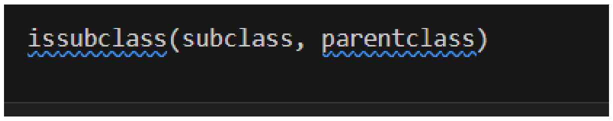
	controlled manner.	“stop()”, and “accelerate()”.	common attributes like “speed()” and methods like “move()”.	would implement the “draw()” method differently, but each could be used interchangeably through the “shape” interface.

- How will you check if a class is a child of another class?

  Answer:

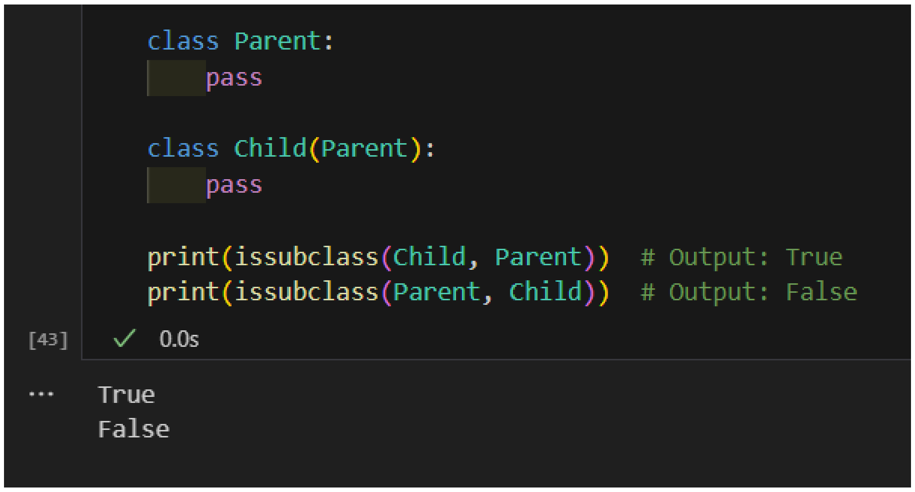
  In Python, the "issubclass()" method can be used to determine whether a class is a child (subclass) of another class. The class you wish to check and the possible parent class are the two inputs this function accepts. In case the first class is a subclass of the second class, it yields "True"; if not, it returns "False".

  Syntax:



```
issubclass(subclass, parentclass)
```

Example:



```
class Parent:
    pass

class Child(Parent):
    pass

print(issubclass(Child, Parent))   # Output: True
print(issubclass(Parent, Child))   # Output: False
```

```
[43]  ✓ 0.0s

...  True
     False
```

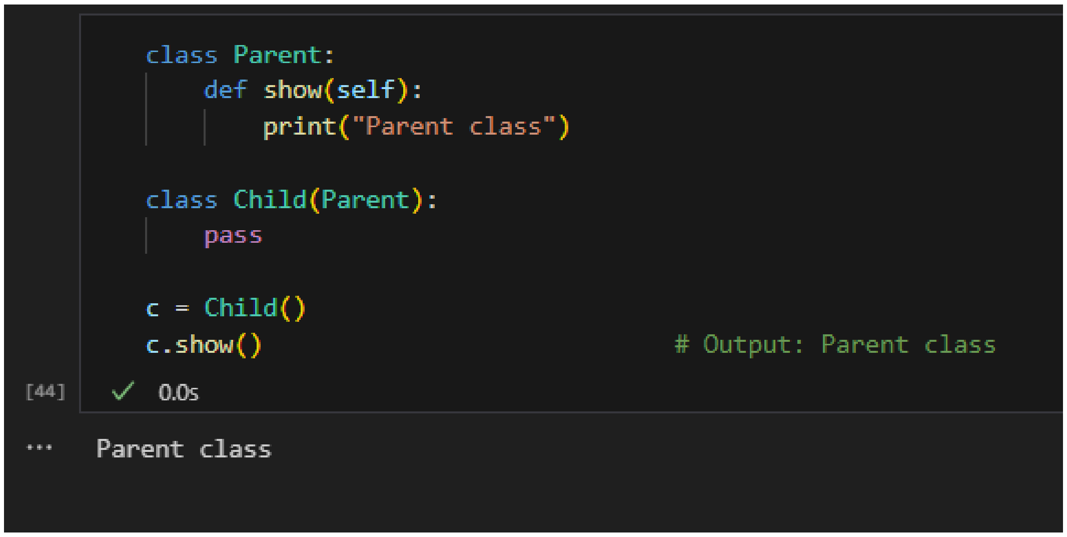
- How does inheritance work in python? Explain all types of inheritance with an example.

  Answer:

  In Python, a class (child or subclass) can inherit properties and methods from another class (parent or superclass) through inheritance. Reusing code and establishing a hierarchical relationship between classes are made possible by this. Python allows for multiple inheritance types:

  - ***Single Inheritance:*** A subclass inherits from one superclass.

    Example:

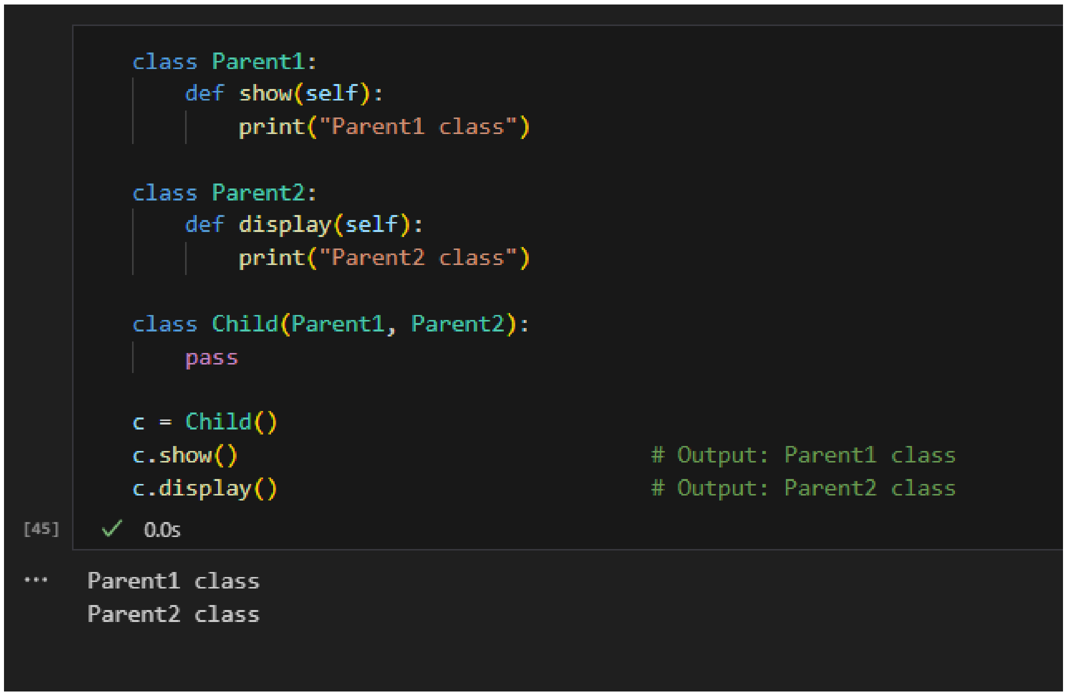


```
class Parent:
    def show(self):
        print("Parent class")

class Child(Parent):
    pass

c = Child()
c.show()                              # Output: Parent class
[44]  ✓ 0.0s
```

```
Parent class
```

  - ***Multiple Inheritance:*** A subclass inherits from multiple superclasses.

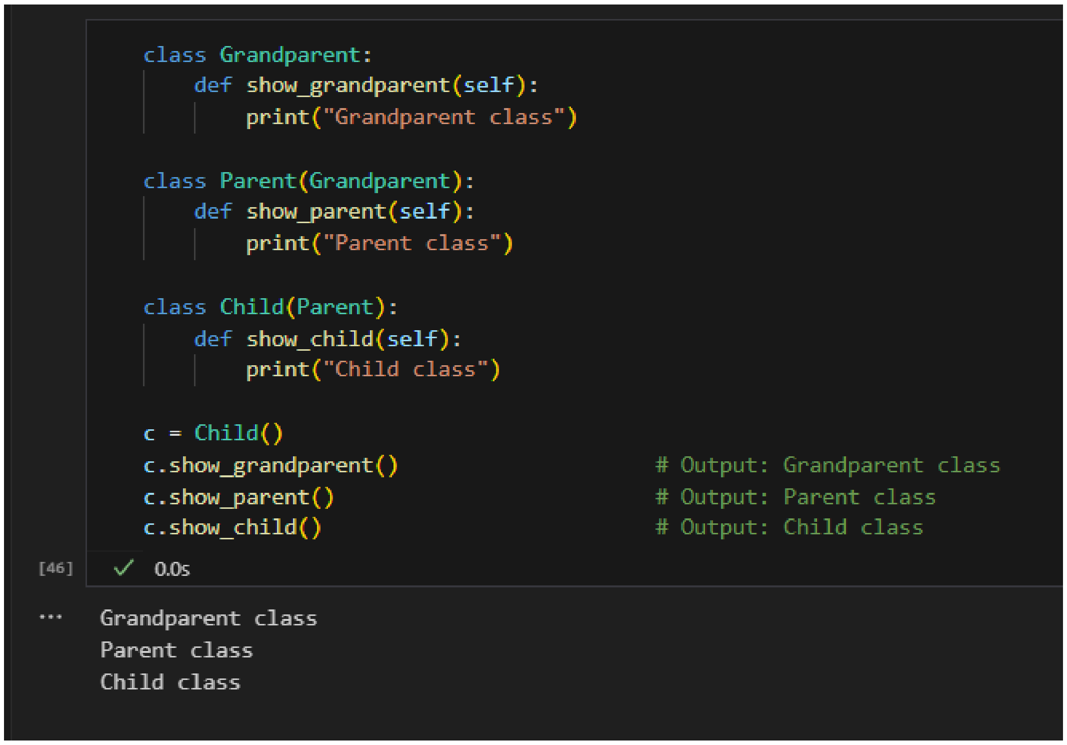
    Example:



```
class Parent1:
    def show(self):
        print("Parent1 class")

class Parent2:
    def display(self):
        print("Parent2 class")

class Child(Parent1, Parent2):
    pass

c = Child()
c.show()                              # Output: Parent1 class
c.display()                           # Output: Parent2 class
[45]  ✓ 0.0s
```

```
Parent1 class
Parent2 class
```

- ***Multilevel Inheritance:*** A subclass inherits from a superclass, which in turn inherits from another superclass.
  Example:

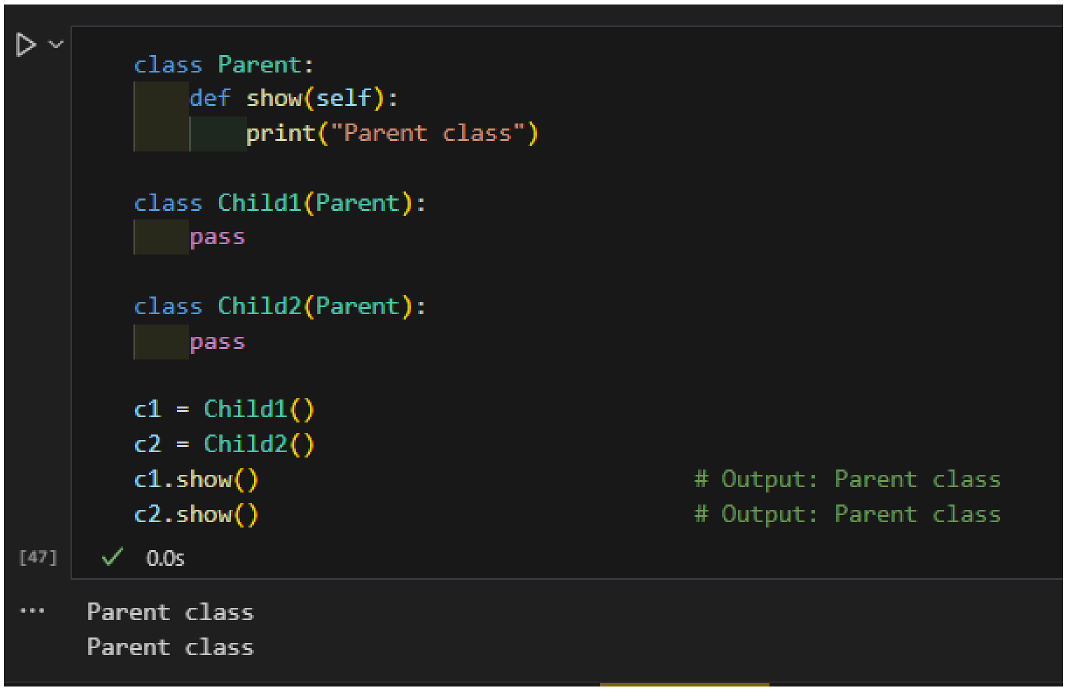


```
class Grandparent:
    def show_grandparent(self):
        print("Grandparent class")

class Parent(Grandparent):
    def show_parent(self):
        print("Parent class")

class Child(Parent):
    def show_child(self):
        print("Child class")

c = Child()
c.show_grandparent()                    # Output: Grandparent class
c.show_parent()                         # Output: Parent class
c.show_child()                          # Output: Child class
```

```
[46] ✓ 0.0s
```

```
Grandparent class
Parent class
Child class
```

- ***Hierarchical Inheritance:*** Multiple subclasses inherit from a single superclass.
  Example:



```
class Parent:
    def show(self):
        print("Parent class")

class Child1(Parent):
    pass

class Child2(Parent):
    pass

c1 = Child1()
c2 = Child2()
c1.show()                               # Output: Parent class
c2.show()                               # Output: Parent class
```

```
[47] ✓ 0.0s
```

```
Parent class
Parent class
```

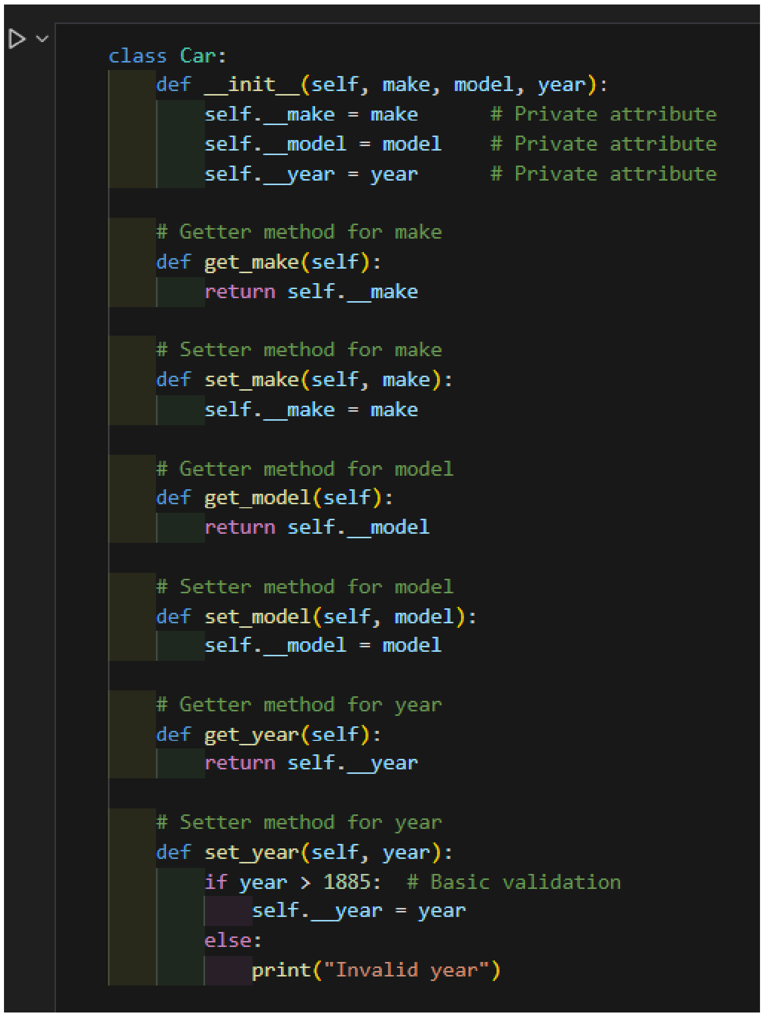
- What is encapsulation? Explain it with an example.

  Answer:

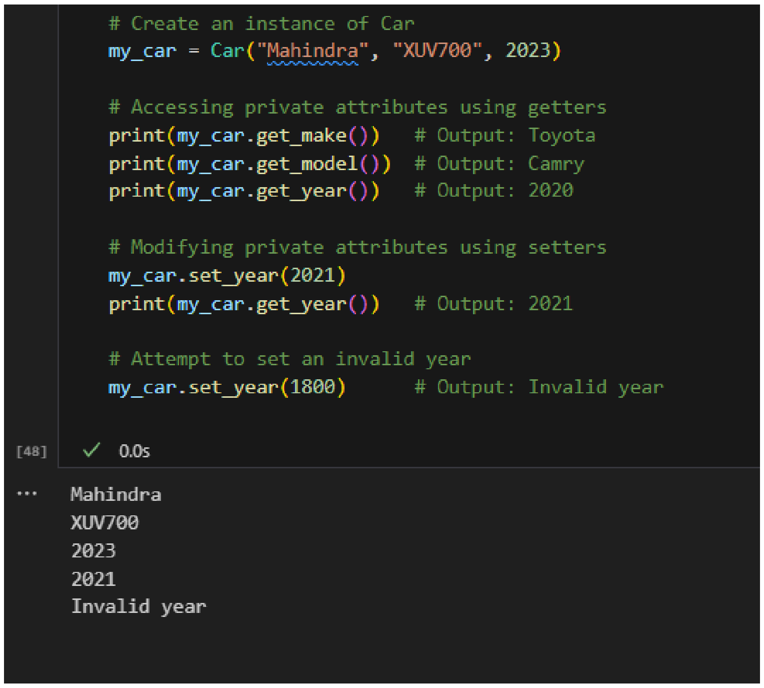
  One of the core ideas of object-oriented programming (OOP) is encapsulation. It describes the combining of methods (functions) that manipulate data and data (attributes) into a single entity, usually a class. Restricting direct access to certain of an object's components is another aspect of encapsulation, which aids in preventing data misuse and unintentional interference. Commonly, access modifiers like private, protected, and public are used to do this. Some of the key concepts of Encapsulation include:-

  - ***Data Hiding:*** Encapsulation hides the internal state of an object from the outside. Only the object's methods can directly interact with its attributes.
  - ***Controlled Access:*** Provides controlled access to the object's data through methods (getters and setters).
  - ***Modularity:*** Encapsulation promotes modularity by keeping the internal workings of an object hidden and only exposing a public interface.

Example:



```python
class Car:
    def __init__(self, make, model, year):
        self.__make = make      # Private attribute
        self.__model = model    # Private attribute
        self.__year = year      # Private attribute

    # Getter method for make
    def get_make(self):
        return self.__make

    # Setter method for make
    def set_make(self, make):
        self.__make = make

    # Getter method for model
    def get_model(self):
        return self.__model

    # Setter method for model
    def set_model(self, model):
        self.__model = model

    # Getter method for year
    def get_year(self):
        return self.__year

    # Setter method for year
    def set_year(self, year):
        if year > 1885:  # Basic validation
            self.__year = year
        else:
            print("Invalid year")
```



```python
# Create an instance of Car
my_car = Car("Mahindra", "XUV700", 2023)

# Accessing private attributes using getters
print(my_car.get_make())   # Output: Toyota
print(my_car.get_model())  # Output: Camry
print(my_car.get_year())   # Output: 2020

# Modifying private attributes using setters
my_car.set_year(2021)
print(my_car.get_year())   # Output: 2021

# Attempt to set an invalid year
my_car.set_year(1800)      # Output: Invalid year
```

[48] ✓ 0.0s

```
Mahindra
XUV700
2023
2021
Invalid year
```

Explanation:

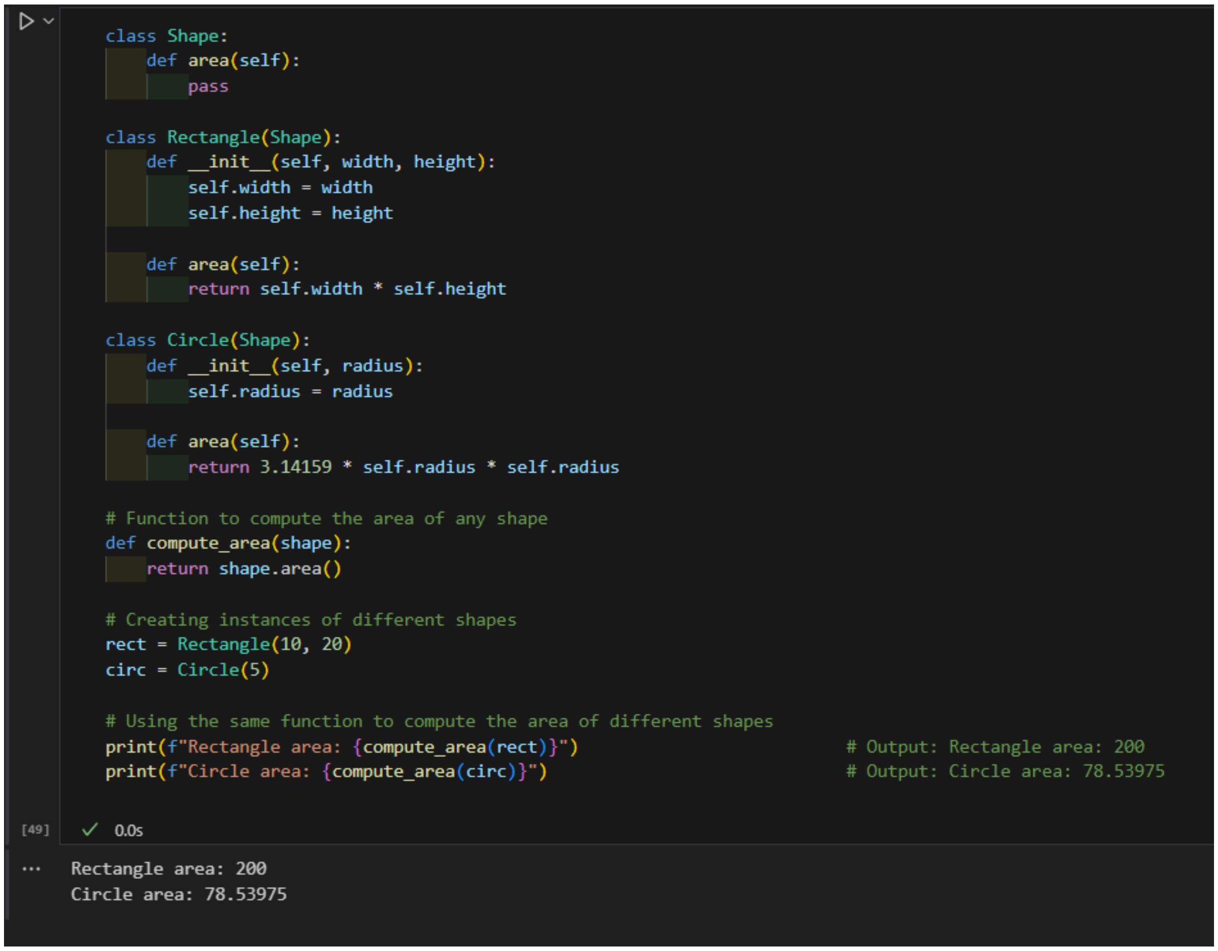
- ***Private Attributes:*** In the "Car" class, the attributes "__make", "__model" and "__year" are private. By convention, double underscores ("__") are used to indicate private attributes that should not be accessed directly from outside the class.
- ***Getter and Setter Methods:*** These methods are used to access and modify the private attributes. This allows for controlled access and modification of the internal state of the object. For example, the "set_year" method includes a basic validation check to ensure the year is valid.
- ***Encapsulation Benefits:***
  - ✓ *Data Hiding:* Direct access to the private attributes is restricted, preventing accidental or unauthorized modifications.
  - ✓ *Controlled Access:* Getters and setters provide a controlled way to access and modify the data, allowing for validation and other logic to be applied.
  - ✓ *Modularity:* The internal implementation of the "Car" class is hidden from the outside, allowing the class to be used as a modular component in larger programs.

- What is polymorphism? Explain it with an example.
  Answer:

The ability to treat objects of distinct classes as objects of a single superclass is made possible by polymorphism, a fundamental idea in object-oriented programming (OOP). It refers to an object's capacity to react differently to an identical method call. The ability to construct methods in a way that works with various data types or classes is what polymorphism does to improve OOP's flexibility and integration. Some of the key concepts include:

- ***Method Overriding:*** Different classes can have methods with the same name, and the correct method is called based on the object's class.
- ***Method Overloading:*** Although not directly supported in Python, it can be achieved using default arguments or by defining methods that can handle different types and numbers of parameters.
- ***Polymorphic Functions:*** Functions that can operate on objects of different types.

Example:



```python
class Shape:
    def area(self):
        pass

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

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

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

    def area(self):
        return 3.14159 * self.radius * self.radius

# Function to compute the area of any shape
def compute_area(shape):
    return shape.area()

# Creating instances of different shapes
rect = Rectangle(10, 20)
circ = Circle(5)

# Using the same function to compute the area of different shapes
print(f"Rectangle area: {compute_area(rect)}")                    # Output: Rectangle area: 200
print(f"Circle area: {compute_area(circ)}")                       # Output: Circle area: 78.53975
```

[49] ✓ 0.0s

```
Rectangle area: 200
Circle area: 78.53975
```

Explanation:

- ***Base Class ("shape"):*** Defines a common interface with an "area" method that will be overridden by subclasses.
- ***Derived Classes ("Rectangle" and "Circle"):*** Both classes inherit from "Shape" and provide their own implementation of the "area" method.
- ***Polymorphic Functions ("compute area"):*** This function takes an object of type "Shape" and calls its "area" method. Thanks to polymorphism, the correct "area" method is called based on the actual object's class.
- ***Method Overriding:*** The "Rectangle" and "Circle" classes override the "area" method from the "Shape" class, providing their specific implementations.

1.2 Which of the following identifier names are invalid and why?

a) Serial_no.
b) 1st_Room
c) Hundred$
d) Total_Marks
e) total-marks
f) Total Marks
g) True
h) _Percentag

Answer:

Identifier Names	Valid / Invalid	Reason
Serial_no.	Invalid	In Python, an identifier cannot end with a period (.). The period / fullstop is not allowed in identifiers.
1st_Room	Invalid	Identifiers cannot start with a digit and in this case, it is starting with 1, which is a digit.
Hundred$	Invalid	The dollar ($) sign is not a valid character for identifiers in Python.
Total_Marks	Valid	This identifier follows all the rules for valid identifiers in Python. It starts with a letter and contains only letters, digits, and underscores.
Total-marks	Invalid	The hyphen "(-)" is not allowed in Python identifiers. Only underscores "(_)" can be used to separate words.
Total Marks	Invalid	Identifiers cannot contain spaces.

True	Invalid	True is a reserved word and reserved words cannot be used for identifiers.
_Percentag	Valid	This identifier starts with an underscore, which is allowed, and contains only letters. It's a valid identifier in Python.

- ***Valid Identifiers:*** “Total_Marks”, “_Percentag”
- ***Invalid Identifiers:*** “Serial_no.”, “1st_Room”, “Hundred$”, “total-marks”, “Total Marks”, “True”.

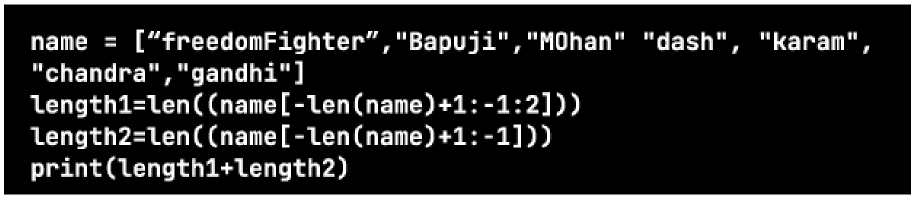
1.3 Answer the following questions:*

name = ["Mohan", "dash", "karam", "chandra", "gandhi", "Bapu"]

Perform the following operations in this list:

a) Add an element "freedom_fighter" in this list at the 0th index.

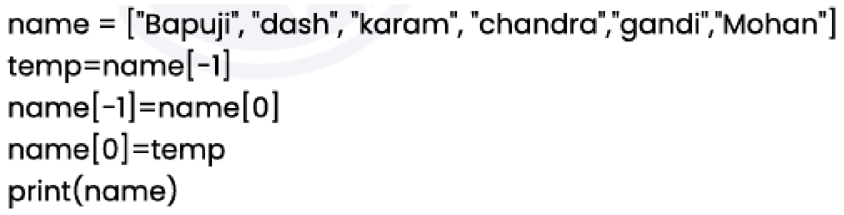
b) Find the output of the following and explain how?



```
name = [“freedomFighter”,"Bapuji","MOhan" "dash", "karam",
"chandra","gandhi"]
length1=len((name[-len(name)+1:-1:2]))
length2=len((name[-len(name)+1:-1]))
print(length1+length2)
```

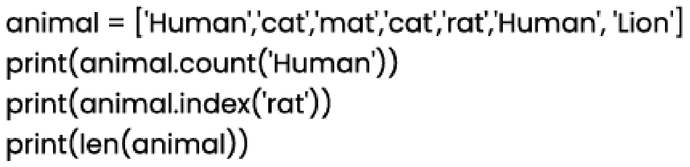
c) Add two more elements in the name ["NetaJi", "Bose"] at the end of the list.

d) What will be the value of temp?



```
name = ["Bapuji", "dash", "karam", "chandra","gandi","Mohan"]
temp=name[-1]
name[-1]=name[0]
name[0]=temp
print(name)
```

1.4 Find the output of the following:*



```
animal = ['Human','cat','mat','cat','rat','Human', 'Lion']
print(animal.count('Human'))
print(animal.index('rat'))
print(len(animal))
```

1.5 Answer the following:*

tuple1 = (10, 20, "Apple", 3.4, 'a', ["master", "ji"], ("sita", "geeta", 22), [{"roll_no" : 1}, {"name" : "Navneet"}])

a) print (len (tuple1))

b) print (tuple1 [-1][-1]["name"])

c) fetch the value of **roll_no** from this tuple

d) print (tuple1 [-3][1])

e) fetch the element "**22**" from this tuple.

1.6 Write a program to display the appropriate message as per the color of signal (RED – Stop / Yellow – Stay / Green - Go) at the road crossing.*

1.7 Write a program to create a simple calculator performing only four basic operations (+, -, /, *)*

1.8 Write a program to find the larger of the three pre-specified numbers using ternary operators.*

1.9 Write a program to find the factors of a whole number using a while loop.*

1.10 Write a program to find the sum of all the positive numbers entered by the user. As soon as the user enters a negative number, stop taking in any further input from the user and display the sum.*

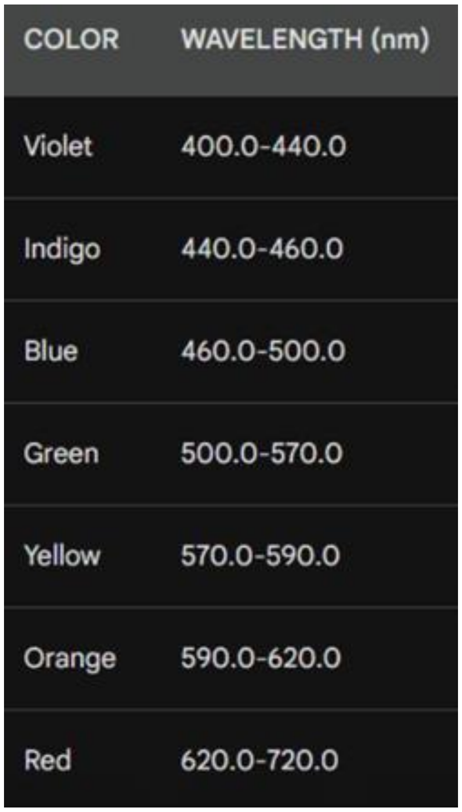
1.11 Write a program to find prime numbers between 2 to 100 using nested for loops.*

1.12 Write the programs for the following:*

- Accept the marks of the student in five major subjects and display the same.
- Calculate the sum of the marks of all subjects. Divide the total marks by number of subjects (i.e. 5), calculate percentage = total marks / 5 and display the percentage.
- Find the grade of the student as per the following criteria.
  Hint: Use Match & case for this.

Criteria	Grade
percentage > 85	A
percentage < 85 && percentage >= 75	B
percentage < 75 && percentage >= 50	C
percentage > 30 && percentage <= 50	D
percentage <30	Reappear

1.13 Write a program for VIBGYOR Spectrum based on their Wavelength using. Wavelength Range:*



COLOR	WAVELENGTH (nm)
Violet	400.0-440.0
Indigo	440.0-460.0
Blue	460.0-500.0
Green	500.0-570.0
Yellow	570.0-590.0
Orange	590.0-620.0
Red	620.0-720.0

1.14 Consider the gravitational interactions between the Earth, Moon, and Sun in our solar system.*

Given,

```
mass_earth = 5.972e24           # Mass of Earth in kilograms
mass_moon = 7.34767309e22      # Mass of Moon in kilograms
mass_sun = 1.989e30            # Mass of Sun in kilograms
distance_earth_sun = 1.496e11   # Average distance between Earth and Sun in meters
distance_moon_earth = 3.844e8  # Average distance between Moon and Earth in meters
```

Tasks:

- Calculate the gravitational force between the Earth and the Sun.
- Calculate the gravitational force between the Moon and the Earth.
- Compare the calculated forces to determine which gravitational force is stronger.
- Explain which celestial body (Earth or Moon) is more attracted to the other based on the comparison.

Solutions:

The questions from 1.3 to 1.14 are coding questions (marked by a * beside the questions) and are solved in a single Jupyter Notebook which has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Question1_Codes.ipynb>

Question No. – 2:

Design and implement a Python program for managing student information using object-oriented principles. Create a class called `Student` with encapsulated attributes for name, age, and roll number. Implement getter and setter methods for these attributes. Additionally, provide methods to display student information and update student details.*

Tasks:

- Define the `Student` class with encapsulated attributes.
- Implement getter and setter methods for the attributes.
- Write methods to display student information and update details.
- Create instances of the `Student` class and test the implemented functionality.

Question No. – 3:

Develop a Python program for managing library resources efficiently. Design a class named `LibraryBook` with attributes like book name, author, and availability status. Implement methods for borrowing and returning books while ensuring proper encapsulation of attributes.*

Tasks:

- Create the `LibraryBook` class with encapsulated attributes.
- Implement methods for borrowing and returning books.
- Ensure proper encapsulation to protect book details.
- Test the borrowing and returning functionality with sample data.

Question No. – 4:

Create a simple banking system using object-oriented concepts in Python. Design classes representing different types of bank accounts such as savings and checking. Implement methods for deposit, withdraw, and balance inquiry. Utilize inheritance to manage different account types efficiently.*

Tasks:

- Define base class(es) for bank accounts with common attributes and methods.
- Implement subclasses for specific account types (e.g., SavingsAccount, CheckingAccount).
- Provide methods for deposit, withdraw, and balance inquiry in each subclass.
- Test the banking system by creating instances of different account types and performing transactions.

Question No. – 5:

Write a Python program that models different animals and their sounds. Design a base class called `Animal` with a method `make_sound()`. Create subclasses like `Dog` and `Cat` that override the `make_sound()` method to produce appropriate sounds.*

Tasks:

- Define the `Animal` class with a method `make_sound()`.
- Create subclasses `Dog` and `Cat` that override the `make_sound()` method.
- Implement the sound generation logic for each subclass.
- Test the program by creating instances of `Dog` and `Cat` and calling the `make_sound()` method.

Question No. – 6:

Write a code for Restaurant Management System Using OOPS:*

- Create a MenuItem class that has attributes such as name, description, price, and category.
- Implement methods to add a new menu item, update menu item information, and remove a menu item from the menu.
- Use encapsulation to hide the menu item's unique identification number.
- Inherit from the MenuItem class to create a FoodItem class and a BeverageItem class, each with their own specific attributes and methods.

Question No. – 7:

Write a code for Hotel Management System using OOPS:*

- Create a Room class that has attributes such as room number, room type, rate, and availability (private).
- Implement methods to book a room, check in a guest, and check out a guest.
- Use encapsulation to hide the room's unique identification number.
- Inherit from the Room class to create a SuiteRoom class and a StandardRoom class, each with their own specific attributes and methods.

Question No. – 8:

Write a code for Fitness Club Management System using OOPS:*

- Create a Member class that has attributes such as name, age, membership type, and membership status (private).
- Implement methods to register a new member, renew a membership, and cancel a membership.
- Use encapsulation to hide the member's unique identification number.
- Inherit from the Member class to create a FamilyMember class and an IndividualMember class, each with their own specific attributes and methods.

Question No. – 9:

Write a code for Event Management System using OOPS:*

- Create an Event class that has attributes such as name, date, time, location, and list of attendees (private).
- Implement methods to create a new event, add or remove attendees, and get the total number of attendees.
- Use encapsulation to hide the event's unique identification number.
- Inherit from the Event class to create a PrivateEvent class and a PublicEvent class, each with their own specific attributes and methods.

Question No. – 10:

Write a code for Airline Reservation System using OOPS:*

- Create a Flight class that has attributes such as flight number, departure and arrival airports, departure and arrival times, and available seats (private).
- Implement methods to book a seat, cancel a reservation, and get the remaining available seats.
- Use encapsulation to hide the flight's unique identification number.
- Inherit from the Flight class to create a DomesticFlight class and an InternationalFlight class, each with their own specific attributes and methods.

Solutions:

The questions from Question No. – 2 to Question No. - 10 are coding questions, based on the concepts of OOPs, which are marked by a * beside the questions. The questions have been solved in a single Jupyter Notebook which has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/OOPS_Codes.ipynb>

Question No. – 11:

Define a Python module named constants.py containing constants like pi and the speed of light.*

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/constants.py>

Question No. – 12:

Write a Python module named calculator.py containing functions for addition, subtraction, multiplication, and division.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/calculator.py>

Question No. – 13:

Implement a Python package structure for a project named ecommerce, containing modules for product management and order processing.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/ecommerce

---

Question No. – 14:

Implement a Python module named string_utils.py containing functions for string manipulation, such as reversing and capitalizing strings.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/string_utils.py>

Question No. – 15:

Write a Python module named file_operations.py with functions for reading, writing, and appending data to a file.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/file_operations.py>

Question No. – 16:

Write a Python program to create a text file named "employees.txt" and write the details of employees, including their name, age, and salary, into the file.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/employees.py>

Question No. – 17:

Develop a Python script that opens an existing text file named "inventory.txt" in read mode and displays the contents of the file line by line.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/inventory.py>

Question No. – 18:

Create a Python script that reads a text file named "expenses.txt" and calculates the total amount spent on various expenses listed in the file.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/expenses.py>

Question No. – 19:

Create a Python program that reads a text file named "paragraph.txt" and counts the occurrences of each word in the paragraph, displaying the results in alphabetical order.

Solutions:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Working_with_Files_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Working_with_Files_Codes/paragraph.py>

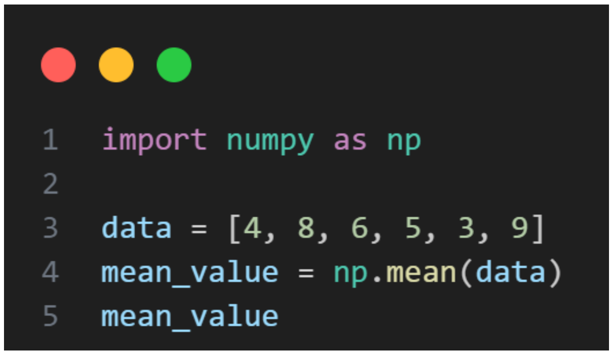
Question No. – 20:

What do you mean by Measure of Central Tendency and Measures of Dispersion? How it can be calculated?

Answer:

Measure of Central Tendency are statistical metrics that explain the center or typical value of a dataset. The most common measures include:-

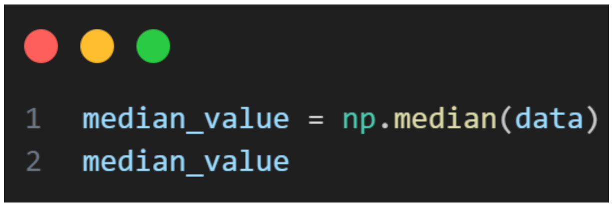
- ***Mean:*** This refers to the average of all data points.
  - ✓ *Calculation:* It is calculated as Mean = $(\Sigma X_i)/N$,
    where,
    X_i is each data point and
    N is the number of data points.
  - ✓ *Example:* Given, a dataset = [4, 8, 6, 5, 3, 9], the mean is:
    Mean = (4 + 8 + 6 + 5 + 3 + 9)/6 = 35/6 = 5.83
  - ✓ *Python Implementation:*



```python
import numpy as np

data = [4, 8, 6, 5, 3, 9]
mean_value = np.mean(data)
mean_value
```

Here also, the answer remains the same.

- ***Median:*** The middle value in an ordered dataset, is known as median.
  - ✓ *Calculation:* Arrange the data in ascending order, and the median is the middle value if N is odd, or the average of the two middle values if N is even.
  - ✓ *Example:* Considering the above example, the dataset is [4, 8, 6, 5, 3, 9] and when arranged in ascending order, it becomes [3, 4, 5, 6, 8, 9]. The median is:
    Median = (5 + 6)/2 = 5.5
  - ✓ *Python Implementation:*



```python
median_value = np.median(data)
median_value
```

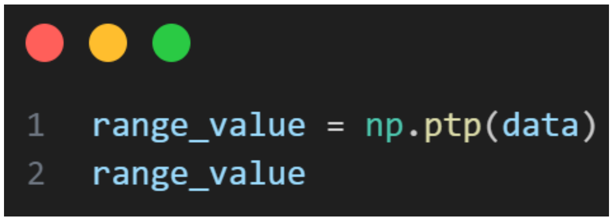
Here also, the answer remains the same.

- ***Mode:*** Mode refers to the most frequently occurring element in the dataset. A dataset may have one mode, more than one mode, or no mode at all.
  - ✓ *Calculation:* Compare the frequency of occurrence of each element in a dataset and identify the element(s) with the most frequency.
  - ✓ *Example:* Considering the above example, where each element occur only once, there is no Mode. But, if one of the number is repeated, then that number will become the mode.

Measures of Dispersion refers to those metrics that explain the spread or variability of the data around the central tendency. The most common measures include:-

- ***Range:*** The difference between the maximum and minimum values.
  - ✓ *Calculation:* Range = Max – Min
  - ✓ *Example:* Considering the above example, the range is:

    Range = (9 – 3) = 6
  - ✓ *Python Implementation:*

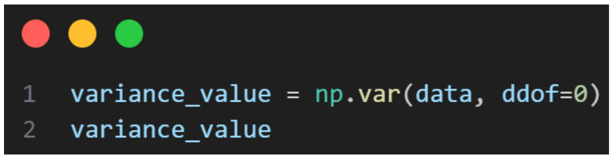


```
range_value = np.ptp(data)
range_value
```

Here also, the answer remains the same.

- ***Variance:*** Variance refers to the average of squared differences from the mean. It measures the spread of values in a dataset.
  - ✓ *Calculation:* Variance = $(\Sigma(X_i - \text{Mean})^2)/N$
  - ✓ *Example:* Considering the above example, the variance is:

    Variance = $((4 - 5.83)^2 + (8 - 5.83)^2 + (6 - 5.83)^2 + (5 - 5.83)^2 + (3 - 5.83)^2 + (9 - 5.83)^2)/6 = 4.47$
  - ✓ *Python Implementation:*

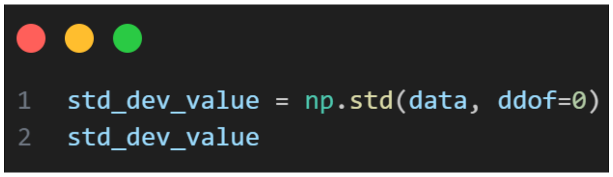


```
variance_value = np.var(data, ddof=0)
variance_value
```

Here also the answer remain the same.

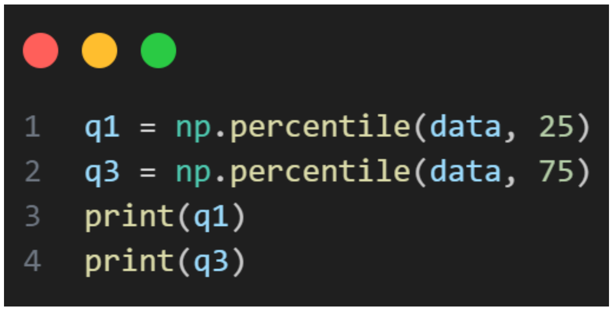
- ***Standard Deviation:*** This is the square root of Variance. It measures the amount of variation or dispersion of a set of values.
  - ✓ *Calculation:* Standard Deviation = $\sqrt{Variance}$
  - ✓ *Example:* Considering the above example, the standard deviation is:

    Standard Deviation = $\sqrt{4.47} = 2.11$
  - ✓ *Python Implementation:*


```
std_dev_value = np.std(data, ddof=0)
std_dev_value
```

Here also the answer remains the same.

- ***Interquartile Range (IQR):*** The range of the middle 50% of the data, is referred to as Interquartile Range (IQR).
  - ✓ *Calculation:* Interquartile Range (IQR) = Q3 – Q1,
    where,
    Q3 is the Third Quartile (75th percentile)
    Q1 is the First Quartile (25th percentile)
  - ✓ *Example:* Considering the above example, the Interquartile Range is:
    Interquartile Range (IQR) = Q3 – Q1 = 7.5 – 4.25 = 3.25
    (since, Q1 = 4.25 and Q3 = 7.5)
  - ✓ *Python Implementation:*


```
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
print(q1)
print(q3)
```

Here also the answer is the same.

Question No. – 21:

What do you mean by skewness? Explain its types. Use graph to show.

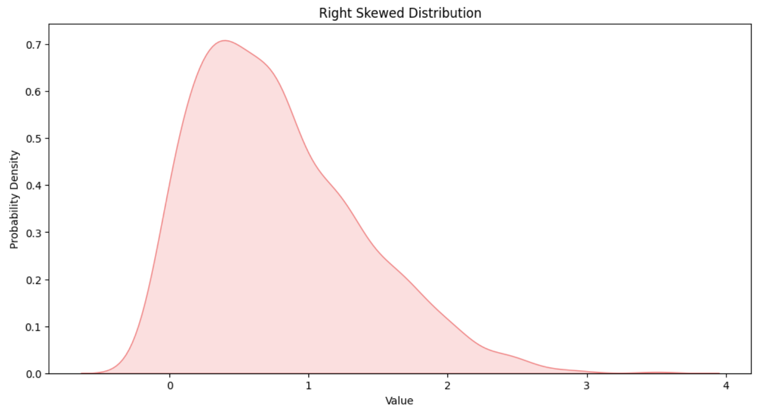
Answer:

Skewness measures the asymmetry of the probability distribution of a real-valued random variable about its mean. When a dataset is skewed, it means that the data points are not symmetrically distributed around the mean. It can tell about the direction and extent to which a distribution deviates from the normal distribution. Skewness can be either positive, negative, or zero, depending on the direction and degree of asymmetry.

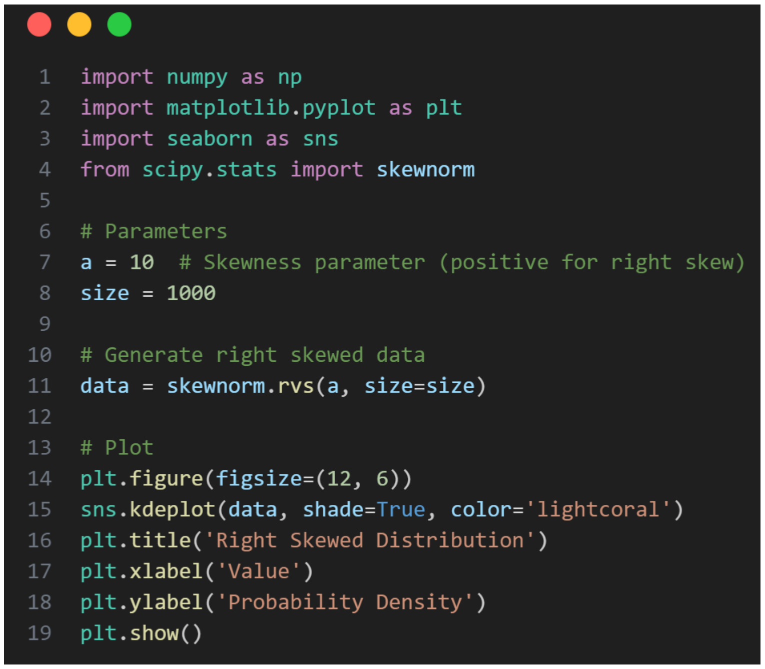
The following are the types of skewness, explained along with graphs:-

- ***Positive Skewness:*** In a positively skewed distribution, the tail on the right side of the distribution is longer or fatter than the left side. Most of the data values are concentrated on the left, but there are a few large values on the right that pull the mean to the right.

- ✓ Mean > Median > Mode
- ✓ The distribution has a long tail on the right.
- ✓ Most data points are concentrated on the left.
- ✓ *Example:* Income distribution in many populations, where most people earn around the median income, but a few individuals earn very high incomes, causing a right-skewed distribution.
- ✓ *Graph:* The graph would show a peak on the left and a long tail stretching out to the right.



- ✓ *Python Implementation:*

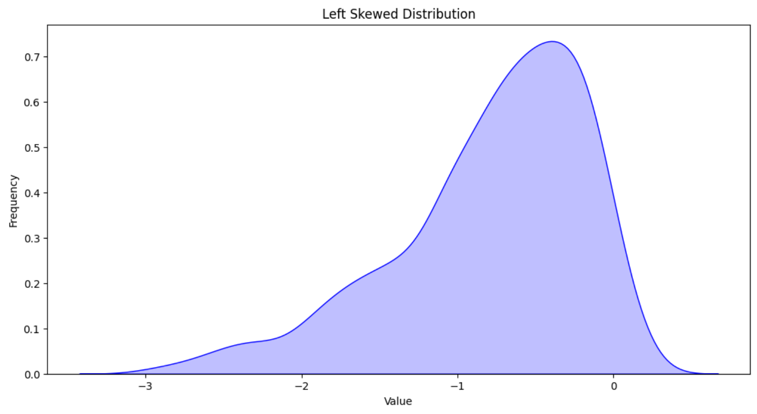


```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skewnorm

# Parameters
a = 10  # Skewness parameter (positive for right skew)
size = 1000

# Generate right skewed data
data = skewnorm.rvs(a, size=size)

# Plot
plt.figure(figsize=(12, 6))
sns.kdeplot(data, shade=True, color='lightcoral')
plt.title('Right Skewed Distribution')
plt.xlabel('Value')
plt.ylabel('Probability Density')
plt.show()
```

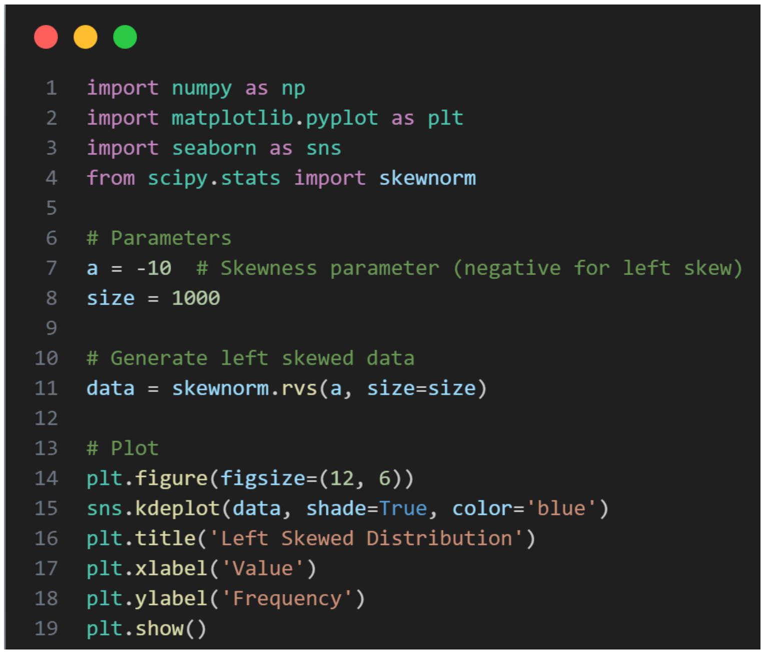
➢ ***Negative Skewness:*** In a negatively skewed distribution, the tail on the left side is longer or fatter than the right side. Most of the data values are concentrated on the right, but there are a few small values on the left that pull the mean to the left.

- ✓ Mean < Median < Mode
- ✓ The distribution has a long tail on the left.

- ✓ Most data points are concentrated on the right.
- ✓ *Example:* Scores on an easy exam, where most students score high marks, but a few score significantly lower, resulting in a left-skewed distribution.
- ✓ *Graph:* The graph would show a peak on the right and a long tail stretching out to the left.

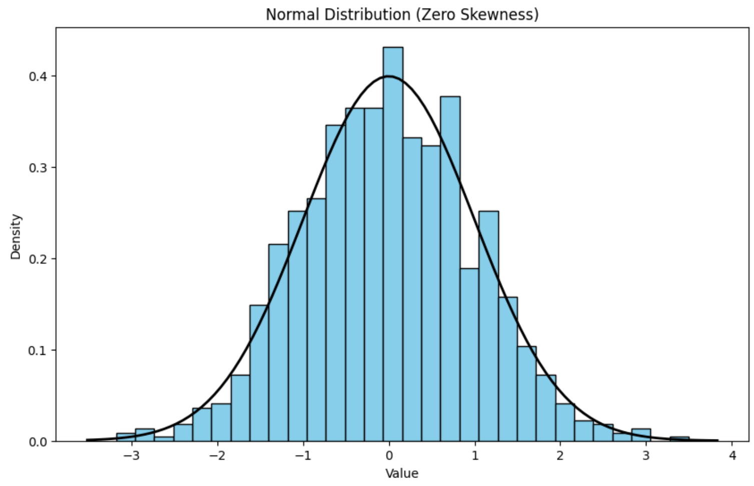


- ✓ *Python Implementation:*

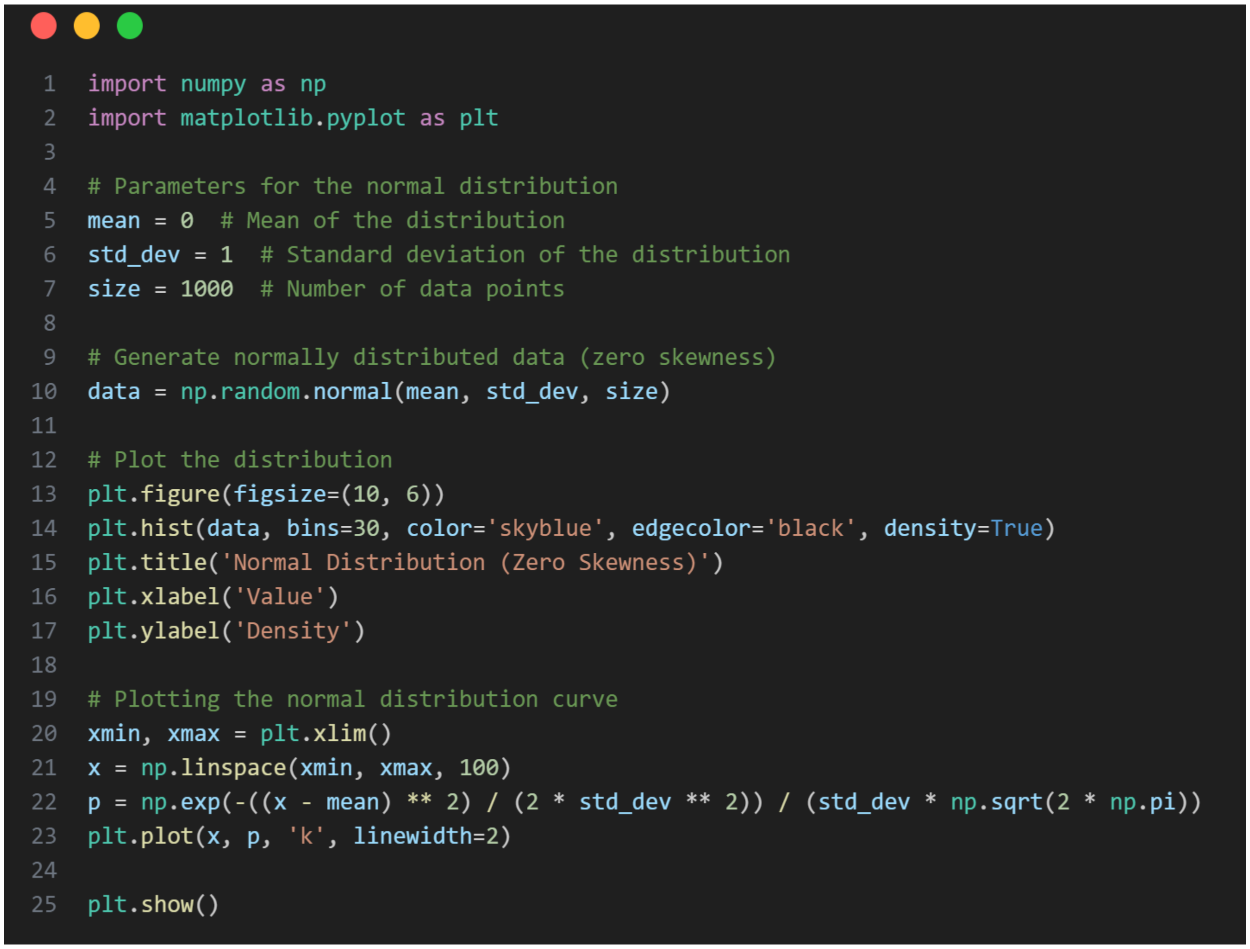


```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skewnorm

# Parameters
a = -10  # Skewness parameter (negative for left skew)
size = 1000

# Generate left skewed data
data = skewnorm.rvs(a, size=size)

# Plot
plt.figure(figsize=(12, 6))
sns.kdeplot(data, shade=True, color='blue')
plt.title('Left Skewed Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

- ➢ ***Zero Skewness:*** In a distribution with zero skewness, the data is symmetrically distributed around the mean. This means the left and right sides of the distribution are mirror images of each other.
  - ✓ Mean = Median = Mode
  - ✓ The distribution has no long tail; both sides taper off equally.
  - ✓ *Example:* The distribution of heights in a large population, where most people have average height and very few are extremely short or tall.
  - ✓ *Graph:* The graph would show a bell-shaped curve, like a normal distribution, with equal tails on both sides.



✓ *Python Implementation:*



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

# Parameters for the normal distribution
mean = 0  # Mean of the distribution
std_dev = 1  # Standard deviation of the distribution
size = 1000  # Number of data points

# Generate normally distributed data (zero skewness)
data = np.random.normal(mean, std_dev, size)

# Plot the distribution
plt.figure(figsize=(10, 6))
plt.hist(data, bins=30, color='skyblue', edgecolor='black', density=True)
plt.title('Normal Distribution (Zero Skewness)')
plt.xlabel('Value')
plt.ylabel('Density')

# Plotting the normal distribution curve
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = np.exp(-((x - mean) ** 2) / (2 * std_dev ** 2)) / (std_dev * np.sqrt(2 * np.pi))
plt.plot(x, p, 'k', linewidth=2)

plt.show()
```

Question No. – 22:

Explain PROBABILITY MASS FUNCTION (PMF) and PROBABILITY DENSITY FUNCTION (PDF) and what is the difference between them?

Answer:

Probability Mass Function (PMF) can be explained as follows:-

- ***Definition of PMF:*** The Probability Mass Function (PMF) is a function that gives the probability that a discrete random variable is exactly equal to some value. It is used for discrete random variables, which are variables that can take on a countable number of distinct values.
- ***Mathematical Definition of PMF:*** For a discrete random variable X with possible values of x_1, x_2, x_3, … , x_n, the PMF is defined as

  $P(X = x_i)$,

  where,

  P represents the Probability.
- ***Properties of PMF:***
  - ✓ $P(X = x_i) \geq 0$ for all x_i
  - ✓ $\Sigma_i P(X = x_i) = 1$, meaning the sum of the probabilities of all possible outcomes equals 1.
  - ✓ Used for discrete random variables.
  - ✓ Describes the probability that a discrete random variable is exactly equal to some value.
- ***Example:*** If a fair six-sided dice is rolled, the PMF for the outcome X is P(X = x) = 1/6P(X = x), for each x in {1, 2, 3, 4, 5, 6}.

Probability Density Function (PDF) can be explained as follows:-

- ***Density of PDF:*** The Probability Density Function (PDF) is a function that describes the likelihood of a continuous random variable to take on a particular value. Unlike the PMF, the PDF is used for continuous random variables, which can take on an infinite number of possible values.
- ***Mathematical Definition of PDF:*** For a continuous random variable X, the PDF is denoted as f(x), where f(x) is the density at point x.
- ***Properties of PDF:***
  - ✓ $f(x) \geq 0$ for all x.
  - ✓ The total area under the PDF curve is 1: $\int_{-\infty}^{\infty} f(x)dx = 1$
  - ✓ The probability that X falls within a particular interval [a, b] is given by the integral of the PDF over that interval: $P(a \leq X \leq b) = \int_a^b f(x)dx$
  - ✓ Used for continuous random variables.

- ✓ Describes the likelihood of a random variable falling within a particular range of values, rather than taking any one value.
- ➢ ***Example:*** For a standard normal distribution (mean = 0, standard deviation = 1), the PDF is $(\frac{1}{\sqrt{2\pi}})$*e$-1/2x^2$

The difference between PMF and PDF are as follows:-

Feature	***PMF***	***PDF***
Type of Variable	PMF is for discrete random variables.	PDF is for continuous random variables.
Representation	Probability for exact values P(X = x)	Probability density f(x)
Cumulative	In PMF, the sum of probabilities for all outcomes equals 1	Integral of the density function and the area under the entire curve, in this case is equal to 1.
Value	PMF gives the actual probability of specific outcomes.	PDF gives a density and the probability is found by integrating the PDF over an interval.

Question No. – 23:

What is correlation? Explain its type in details. What are the methods of determining correlation?

Answer:

Correlation measures the strength and direction of the linear relationship between two variables. It quantifies how changes in one variable are associated with changes in another.

The types of correlations are as follows:-

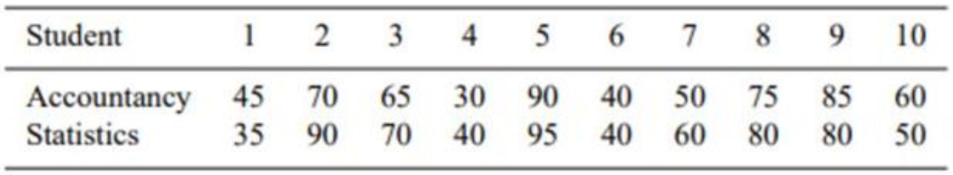
- ***Positive Correlation:***
  - ✓ As one variable increases, the other variable also increases.
  - ✓ *Example:* Height and weight.
- ***Negative Correlation:***
  - ✓ As one variable increases, the other variable decreases.
  - ✓ *Example:* Speed of a car and travel time.
- ***Zero Correlation:***
  - ✓ No linear relationship between the variables.
  - ✓ *Example:* Shoe size and intelligence.

The methods of determining correlation are as follows:-

- ***Pearson Correlation Coefficient:***
  - ✓ Measures linear correlation between two variables.
  - ✓ Value ranges from -1 to 1, where – 1 indicates a perfect positive correlation; -1 indicates a perfect negative correlation and 0 indicates no correlation.
  - ✓ *Calculation:* $r = \sum(Xi - X')(Yi - Y')/\sqrt{}\sum(Xi - X')2\sum(Yi - Y')2$
- ***Spearman's Rank Correlation:***
  - ✓ Non-parametric measure of correlation based on ranks of data.
  - ✓ Useful for ordinal data or when the assumptions of Pearson's correlation are not met.
  - ✓ *Calculation:* $r_s = (1 - 6\sum d_i^2)/(n(n^2-1))$
    where,
    d_i is the difference between ranks.
- ***Kendall's Tau:***
  - ✓ Measures the association between two measured quantities.
  - ✓ Used when the dataset is small or has many tied ranks.
  - ✓ *Calculation:* Based on the number of concordant and discordant pairs of observations.

Question No. – 24:

Calculate coefficient of correlation between the marks obtained by 10 students in Accountancy and Statistics: #



Student	1	2	3	4	5	6	7	8	9	10
Accountancy	45	70	65	30	90	40	50	75	85	60
Statistics	35	90	70	40	95	40	60	80	80	50

Use Karl Pearson's Coefficient of Correlation Method to find it.

Answer:

As per the given problem,

Student	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Accountancy	45	70	65	30	90	40	50	75	85	60
Statistics	35	90	70	40	95	40	60	80	80	50

Let us assume,

X = Marks in Accountancy

Y = Marks in Statistics

Step 1: Calculate the Mean of X and Y:

X_{mean} = X' = (45 + 70 + 65 + 30 + 90 + 40 + 50 + 75 + 85 + 60)/10 = 610/10 = 61

Y_{mean} = Y' = (35 + 90 + 70 + 40 + 95 + 40 + 60 + 80 + 80 + 50)/10 = 640/10 = 64

Step 2: Calculate the Deviations (X – X') and (Y – Y') and their Corresponding Sums:

Student	***X***	***X'***	***(X – X')***	***$(X – X')^2$***	***Y***	***Y'***	***(Y – Y')***	***$(Y – Y')^2$***	***(X – X')(Y – Y')***
1	45	61	-16	256	35	64	-29	841	464
2	70	61	9	81	90	64	26	676	234
3	65	61	4	16	70	64	6	36	24
4	30	61	-31	961	40	64	-24	576	744
5	90	61	29	841	95	64	31	961	899
6	40	61	-21	441	40	64	-24	576	504
7	50	61	-11	121	60	64	-4	16	44
8	75	61	14	196	80	64	16	256	224
9	85	61	24	576	80	64	16	256	384
10	60	61	-1	1	50	64	-14	196	14
Total	610	610	0	3490	640	640	0	4384	3536

Step 3: Calculate the Coefficient of Correlation r:

We know,

$$r = \sum(Xi - X')(Yi - Y')/\sqrt{}\sum(Xi - X')2\sum(Yi - Y')2$$

On substituting the values, we get:

$$r = \frac{3536}{\sqrt{(3490)*(4384)}} = 0.904$$

Ans.:- The Coefficient of Correlation is r = 0.904

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 25:

Discuss the 4 differences between Correlation and Regression.

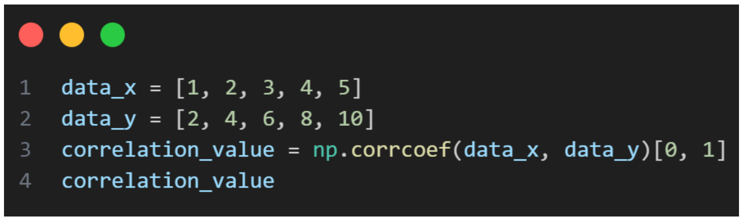
Answer:

Four key differences between Correlation and Regression are as follows:-

Properties	***Correlation***	***Regression***
Purpose	Measures the strength and direction of the linear relationship between two variables. It does not imply causality.	Models the relationship between a dependent variable and one or more independent variables, often to predict or estimate the dependent variable based on the independent variables.
Nature of Relationship	Symmetrical; The correlation between X and Y is the same as between Y and X.	Asymmetrical; Regression focuses on predicting Y from X, so reversing the roles of X and Y will lead to different equations.
Dependency	Does not distinguish between dependent and independent variables; it merely quantifies the relationship.	Explicitly distinguishes between dependent (outcome) and independent (predictor) variables, modeling the dependent variable as a function of the independent variables.
Output	Provides a single value called the correlation coefficient (r), which ranges from -1 to 1. The value indicates the strength and direction of the relationship.	Provides an equation, in the form of $Y = a + bX$, where, a is the intercept and b is the slope. It gives a specific predictive model.

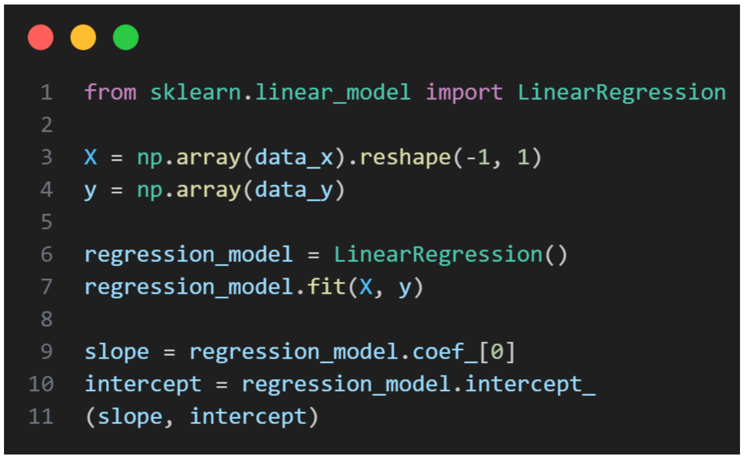
Now, let's take a look at an example each for Correlation and Regression.

- ***Example of Correlation:*** Given two datasets: [1, 2, 3, 4, 5] and [2, 4, 6, 8, 10], the correlation is 1, indicating a perfect positive linear relationship.
  - ✓ *Python Implementation:*



```python
data_x = [1, 2, 3, 4, 5]
data_y = [2, 4, 6, 8, 10]
correlation_value = np.corrcoef(data_x, data_y)[0, 1]
correlation_value
```

- ***Example of Regression:*** Given two datasets: [1, 2, 3, 4, 5] (independent variable) and [2, 4, 6, 8, 10] (dependent variable), the linear regression equation is:
  Y = 2x + 0
  - ✓ *Python Implementation:*



```python
from sklearn.linear_model import LinearRegression

X = np.array(data_x).reshape(-1, 1)
y = np.array(data_y)

regression_model = LinearRegression()
regression_model.fit(X, y)

slope = regression_model.coef_[0]
intercept = regression_model.intercept_
(slope, intercept)
```

Question No. – 26:

Find the most likely price at Delhi corresponding to the price of Rs. 70 at Agra from the following data: #

Coefficient of correlation between the prices of the two places +0.8.

Answer:

According to the given problem,

Coefficient of correlation (r) between the prices at Delhi and Agra = +0.8

Price at Agra (x_A) = Rs. 70

Let us assume,

Price at Delhi = x_D

Therefore,

The regression equation of x_D and x_A can be written as follows:

$$x_D = x_D' + r * \frac{\sigma D}{\sigma A} * (x_A - x_A')$$

where,

x_D → Predicted Price at Delhi

x_D' → Mean Price at Delhi

x_A → Predicted Price at Agra

x_A' → Mean Price at Agra

r → Correlation Coefficient Between Prices at Agra and Delhi

σD → Standard Deviation of Prices at Delhi

σA → Standard Deviation of Prices at Agra

Since,

The specific values for x_D', x_A', σD and σA are not available, so a general approach to use the linear relationship will be used:

$$x_D = r * x_A$$

Therefore,

$$x_D = 0.8 * 70 = \text{Rs. } 56$$

Ans.:- The most likely price at Delhi is going to be Rs. 56

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assi

gnment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 27:

In a partially destroyed laboratory record of an analysis of correlation data, the following results only are legible: #

Variance of x = 9,

Regression equations are:

(i) $8x - 10y = -66$;
(ii) $40x - 18y = 214$.

What are:

(a) the mean values of x and y,

(b) the coefficient of correlation between x and y,

(c) the σ of y.

Answer:

According to the given problem,

Variance of x = 9

Regression Equations:

(i) $8x - 10y = -66$
(ii) $40x - 18y = 214$

(a) Find the Mean Values of x and y

The equations can be re-written as:

(i) $8x - 10y = -66 \rightarrow y = 0.8x + 6.6$
(ii) $40x - 18y = 214 \rightarrow y = 2.22x - 11.89$

On equating y, we get:

$0.8x + 6.6 = 2.22x - 11.89$

$\rightarrow$ $1.42x = 18.49$

$\rightarrow$ $x = 18.49/1.42$

→ $x = 13.02$

Now,

On putting the value of x in Equation (i) we get:

$y = 0.8 * 13.02 + 6.6$

→ $y = 17.02$

Therefore,

Mean value of x = x' = 13.02

Mean value of y = y' = 17.02

(b) ***Find the Coefficient of Correlation r:***

The regression coefficients b_{yx} and b_{xy} can be calculated as follows:

For the 1st Regression Equation: $8x - 10y = -66$

$b_{yx} = 8/10 = 0.8$

For the 2nd Regression Equation: $40x - 18y = 214$

$b_{xy} = 18/40 = 0.45$

Therefore,

The coefficient of correlation r is given by,

$$r = \sqrt{byx} * bxy = \sqrt{0.8} * 0.45 = \sqrt{0.36} = 0.6$$

(c) ***Find the σ of y:***

Given,

The variance of x is 9,

Therefore,

The Standard Deviation of x is: $\sigma_x = \sqrt{9} = 3$

We know,

The relationship between Regression Coefficients and Standard Deviations:

$b_{yx} = r * \sigma_y/\sigma_x$

On substituting the values, we get:

$0.8 = 0.6 * \sigma_y/3$

→ $\sigma_y = (0.8 * 3)/0.6 = 4$

Ans.:- (a) Mean values: x' = 13.02 and y' = 17.02

(b) Coefficient of Correlation: r = 0.6

(c) Standard Deviation: $\sigma_y = 4$

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 28:

What is Normal Distribution? What are the four Assumptions of Normal Distribution? Explain in detail.

Answer:

Normal Distribution is a continuous probability distribution that is symmetric about the mean, showing that data near the mean are more frequent in occurrence than data far from the mean. In graphical form, the normal distribution will appear as a bell curve.

The probability density function (PDF) of a normal distribution is given by:

$$f(x) = (\frac{1}{\sigma\sqrt{2\pi}})^{*}e{-}(x - \mu)^{\wedge}2/2\sigma^2$$

where,

μ is the mean and σ is the standard deviation.

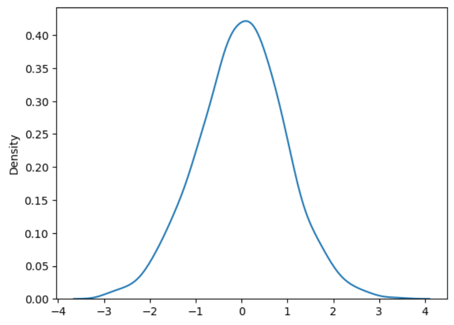
The four assumptions about Normal Distribution along with their explanations are as follows:-

- ***Symmetry:*** The distribution is symmetric about the mean. This means that the left side of the mean is a mirror image of the right side.
- ***Uni - Modality:***
    - ✓ In a normal distribution, the mean, median, and mode are all equal and located at the center of the distribution.
    - ✓ The distribution has a single peak (mode), meaning it is uni-modal. This implies that the most frequent value (mode) occurs at the mean of the distribution.
- ***Bell-Shaped Curve:*** The Normal Distribution has a characteristic bell-shaped curve. It is thin at the tails and thick in the middle, implying that the highest point is at the mean and it tapers off equally on both sides.
- ***Asymptotic Nature:*** The curve approaches the x-axis but never actually touches it, meaning the probability of the variable taking an infinitely large positive or negative value is zero.

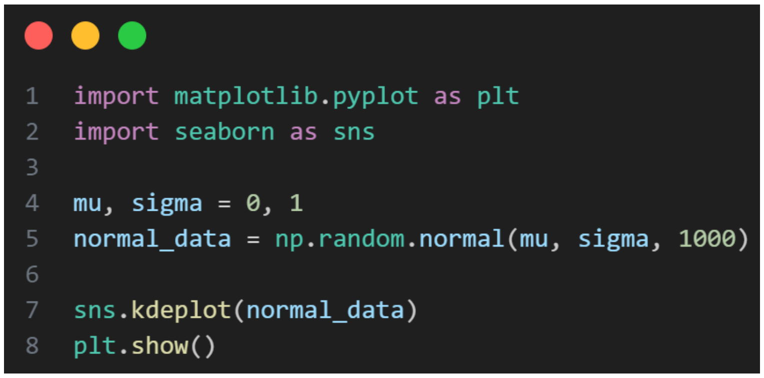
Now, let's take a look at an example of Normal Distribution.

Example:- A dataset with mean = 0 and standard deviation = 1 is normally distributed.

Graph:-



Python Implementation:-



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

mu, sigma = 0, 1
normal_data = np.random.normal(mu, sigma, 1000)

sns.kdeplot(normal_data)
plt.show()
```

Question No. – 29:

Write all the characteristics or properties of the Normal Distribution Curve.

Answer:

The characteristics or properties of Normal Distribution Curve are as follows:-

- ***Symmetry:*** The distribution is symmetric about the mean. This means that the left side of the mean is a mirror image of the right side.
- ***Uni – Modality:***
  - ✓ In a normal distribution, the mean, median, and mode are all equal and located at the center of the distribution.
  - ✓ The distribution has a single peak (mode), meaning it is uni-modal. This implies that the most frequent value (mode) occurs at the mean of the distribution.
- ***Bell – Shaped Curve:*** The Normal Distribution has a characteristic bell-shaped curve. It is thin at the tails and thick in the middle, implying that the highest point is at the mean and it tapers off equally on both sides.
- ***Area Under the Curve:*** The total area under the curve is equal to 1, representing the total probability of all outcomes.
- ***Empirical Rule (68 – 95 – 99.7 Rule):***
  - ✓ Approximately 68% of the data falls within one standard deviation of the mean.
  - ✓ Approximately 95% falls within two standard deviations.
  - ✓ Approximately 99.7% falls within three standard deviations.
- ***Asymptotic Nature:*** The curve approaches the x-axis but never actually touches it, meaning the probability of the variable taking an infinitely large positive or negative value is zero.

Question No. – 30:

Which of the following options are correct about Normal Distribution Curve?

(a) Within a range 0.6745 of σ on both sides the middle 50% of the observations occur i,e. mean ±0.6745σ covers 50% area 25% on each side.

(b) Mean ±1S.D. (i,e. μ ± 1σ) covers 68.268% area, 34.134 % area lies on either side of the mean.

(c) Mean ±2S.D. (i,e. μ ± 2σ) covers 95.45% area, 47.725% area lies on either side of the mean.

(d) Mean ±3 S.D. (i,e. μ ±3σ) covers 99.73% area, 49.856% area lies on the either side of the mean.

(e) Only 0.27% area is outside the range μ ±3σ.

Answer:

All the ***Options (a), (b), (c), (d) and (e)*** are correct.

Justification:

(a) This is correct. ±0.6745σ around the mean covers 50% of the data in a normal distribution.
(b) The empirical rule states that about 68.27% of data lies within one standard deviation from the mean in a normal distribution. So, the given statement, option (b) is correct.
(c) Approximately 95.45% of the data lies within two standard deviations from the mean in a normal distribution. So, the given statement, option (c) is correct.
(d) Around 99.73% of the data falls within three standard deviations from the mean, with approximately 49.865% on each side. So, the given statement, option (d) is also correct.
(e) Since 99.73% of data is within ±3σ, the remaining 0.27% lies outside this range. So, the given statement, option (e) is also correct.

Question No. – 31:

The mean of a distribution is 60 with a standard deviation of 10. Assuming that the distribution is normal, what percentage of items be: #

(i) between 60 and 72,
(ii) between 50 and 60,
(iii) beyond 72 and
(iv) between 70 and 80?

Answer:

According to the given problem,

Mean $\mu = 60$

Standard Deviation $\sigma = 10$

(i) Between 60 and 72:

$Z_{60} = (60 - 60) / 10 = 0$

$Z_{72} = (72 - 60) / 10 = 1.2$

From Z – table, we get the following information:

The area to the left of Z=1.2 is approximately 0.8849.

The area to the left of Z=0 (mean) is 0.5.

Therefore,

The percentage of items between 60 and 72 is:

Percentage = (0.8849 – 0.5) * 100% = 38.49%

(ii) ***Between 50 and 60:***

$Z_{50} = (50 - 60) / 10 = -1$

$Z_{60} = (60 - 60) / 10 = 0$

From Z – table, we get the following information:

The area to the left of Z=-1 is approximately 0.1587.

The area to the left of Z=0 (mean) is 0.5.

Therefore,

The percentage of items between 50 and 60 is:

Percentage = (0.5 - 0.1587.) * 100% = 34.13%

(iii) Beyond 72:

$Z_{72} = (72 - 60) / 10 = 1.2$

The area to the right of Z = 1.2 is:

Percentage = (1 – 0.8849) * 100% = 11.51%

(iv) ***Between 70 and 80:***

$Z_{70} = (70 - 60) / 10 = 1$

$Z_{80} = (80 - 60) / 10 = 2$

From Z – table, we get the following information:

The area to the left of Z = 1 is approximately 0.8413.

The area to the left of Z = 0 (mean) is 0.9772.

Therefore,

The percentage of items between 70 and 80 is:

Percentage = (0.9772 – 0.8413) * 100% = 13.59%

Ans.:- (i) The percentage of items between 60 and 72 is 38.49%

(ii) The percentage of items between 50 and 60 is 34.13%

(iii) The percentage of items beyond 72 is 11.51%

(iv) The percentage of items between 70 and 80 is 13.59%

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 32:

15000 students sat for an examination. The mean marks was 49 and the distribution of marks had a standard deviation of 6. Assuming that the marks were normally distributed what proportion of students scored: #

(a) more than 55 marks,

(b) more than 70 mark

Answer:

According to the given problem,

Number of students = 15000

Mean marks $\mu = 49$

Standard Deviation $\sigma = 6$

(a) Find the proportion of students scoring more than 55 marks:

$Z_{55} = (55 - 49) / 6 = 1$

The area to the right of Z = 1:

Proportion = 1 - 0.8413 = 0.1587

Number of students = 0.1587 * 15000 = 2380.5 ≈ 2380

(b) Find the proportion of students scoring more than 70 marks:

$Z_{70} = (70 - 49) / 6 = 3.5$

The area to the right of Z = 3.5:

Proportion = 1 - 0.9997 = 0.0003

Number of students = 0.0003 * 15000 = 4.5 ≈ 4

Ans.:- (a) Proportion of students scoring more than 55 marks: 2380.5 ≈ 2380

(b) Proportion of students scoring more than 70 marks: 4.5 ≈ 4

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 33:

If the height of 500 students are normally distributed with mean 65 inch and standard deviation 5 inch. How many students have height:

a) greater than 70 inch.

b) between 60 and 70 inch.

Answer:

According to the given problem,

Number of students = 500

Mean height μ = 65 inches

Standard Deviation σ = 5 inches

(a) Find the Number of Students with height greater than 70 inches:

Z_{70} = (70 – 65) / 5 = 1

The area to the right of Z = 1:

Proportion = 1 - 0.8413 = 0.1587

Number of students = 0.1587 * 500 = 79.35 ≈ 79

(b) Find the Number of students with height between 60 inches and 70 inches:

Z_{60} = (60 – 65) / 5 = -1

Z_{70} = (70 – 65) / 5 = 1

The area between Z = 1 and Z = -1:

Proportion = 0.8413 – 0.1587 = 0.6826

Number of students = 0.6826 * 500 = 341.3 ≈ 341

Ans.:- (a) The number of students with height greater than 70 inches are 79.35 ≈ 79

(b) The number of students with height between 60 inches and 70 inches are

341.3 ≈ 341

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 34:

What is statistical hypothesis? Explain the errors in hypothesis testing. Explain the Sample. What are Large Samples & Small Samples?

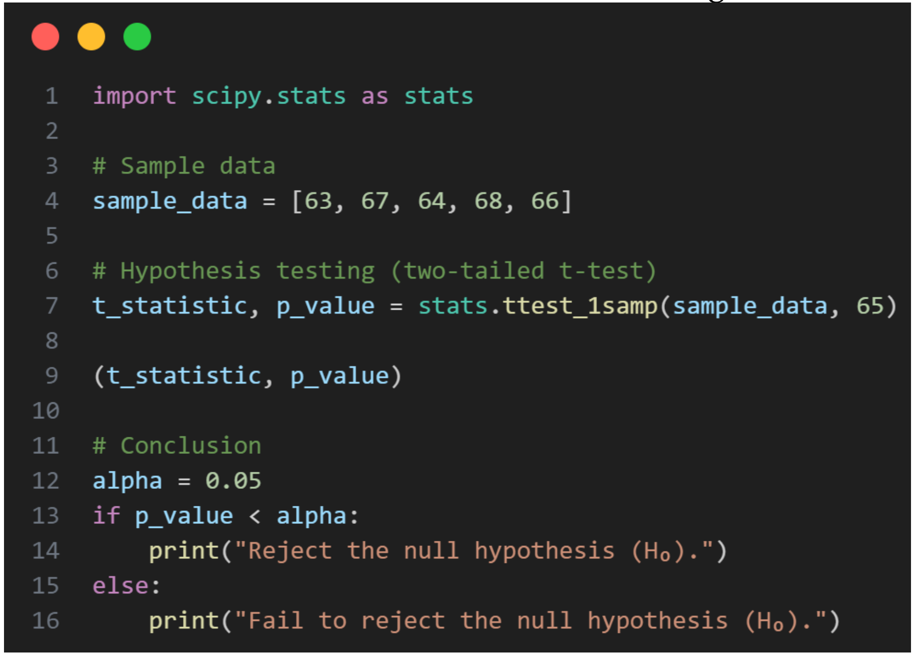
Answer:

A Statistical Hypothesis is an assumption or claim about a population parameter (such as the mean or variance) that can be tested using statistical methods. Hypothesis testing is a procedure for deciding whether the hypothesis should be accepted or rejected based on sample data.

- **Null Hypothesis (H_0):** The hypothesis that there is no effect or no difference, often denoted as H_0. It is the hypothesis that is tested.
- **Alternative Hypothesis (H_1):** The hypothesis that there is an effect or a difference, contrary to the null hypothesis.

Example of Hypothesis: Testing whether the mean height of students is 65 inches.

- H_0: The mean height is 65 inches.
- H_1: The mean height is not 65 inches.
- Python Implementation:



```python
import scipy.stats as stats

# Sample data
sample_data = [63, 67, 64, 68, 66]

# Hypothesis testing (two-tailed t-test)
t_statistic, p_value = stats.ttest_1samp(sample_data, 65)

(t_statistic, p_value)

# Conclusion
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis (H₀).")
else:
    print("Fail to reject the null hypothesis (H₀).")
```

Here, it has failed to reject null hypothesis.

The errors in Hypothesis Testing have been explained as follows:-

- ***Type I Error (α):***
  - ✓ Occurs when the null hypothesis is rejected when it is actually true.
  - ✓ It is also known as a false positive.
  - ✓ The probability of committing a Type I error is denoted by α (significance level).
- ***Type II Error (β):***
  - ✓ Occurs when the null hypothesis is not rejected when it is actually false.
  - ✓ It is also known as a false negative.
  - ✓ The probability of committing a Type II error is denoted by β.

A Sample is a subset of a population used to represent the population in statistical analysis. Samples are used because it is often impractical or impossible to study an entire population.

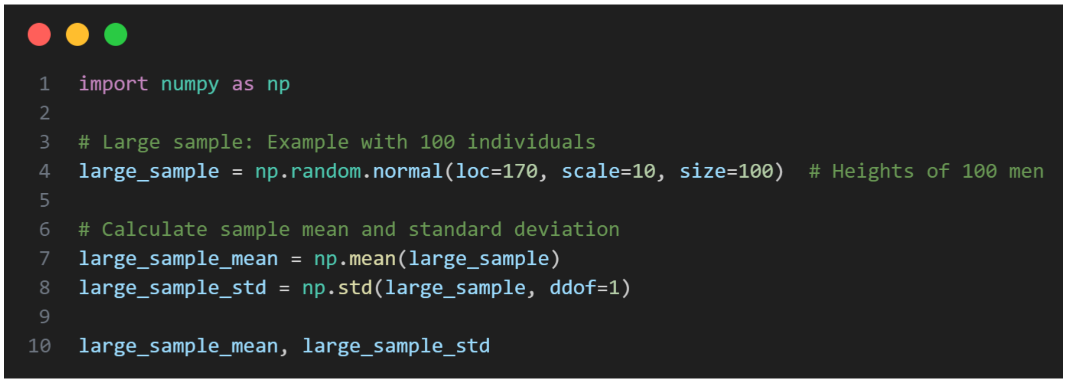
Large Sample: A sample is generally considered large when it contains 30 or more observations. This is a rule of thumb, and the exact threshold can vary depending on the context or the statistical methods being used.

- ***Characteristics of Large Samples:***
  - ✓ *Central Limit Theorem (CLT):* For large samples, the sampling distribution of the sample mean tends to be approximately normally distributed, regardless of the original distribution of the population. This allows for the use of parametric tests, which assume normality.

- ✓ *Reliability:* Large samples tend to provide more accurate and reliable estimates of population parameters (such as the mean or proportion) because they reduce the effect of random sampling errors.
- ✓ *Statistical Tests:* In large samples, standard statistical tests like the Z-test for means and proportions can be used with greater confidence in their results.

➢ ***Example:*** If a researcher is studying the average height of adult men in a city and collects data from 1000 individuals, this would be considered a large sample.

➢ ***Python Implementation:***



```python
import numpy as np

# Large sample: Example with 100 individuals
large_sample = np.random.normal(loc=170, scale=10, size=100)  # Heights of 100 men

# Calculate sample mean and standard deviation
large_sample_mean = np.mean(large_sample)
large_sample_std = np.std(large_sample, ddof=1)

large_sample_mean, large_sample_std
```

The output is: (170.23505683851897, 10.049886949045804)

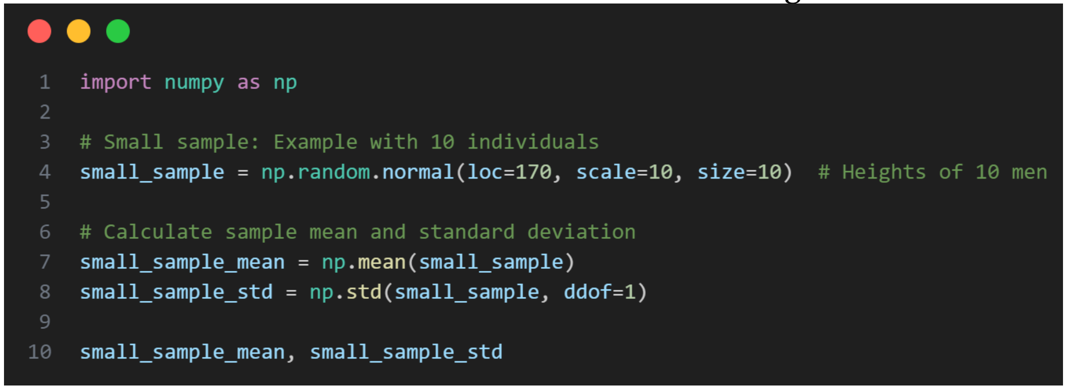
Small Sample: A sample is considered small when it contains fewer than 30 observations. This is again a general guideline, and the exact threshold may vary.

➢ ***Characteristics of Small Sample:***

- ✓ *Less – Reliable Estimates:* Small samples are more susceptible to the influence of outliers or random variations, which can lead to less reliable estimates of population parameters.
- ✓ *Non – Normal Distributions:* The sampling distribution of the sample mean may not be normally distributed, especially if the population itself is not normal. Therefore, different techniques, such as the t-distribution, are often used for hypothesis testing.
- ✓ *Statistical Tests:* For small samples, the t-test is often used instead of the Z-test, particularly when the population variance is unknown.

➢ ***Example:*** If a researcher is studying a rare disease and only has data from 15 patients, this would be considered a small sample.

➢ ***Python Implementation:***



```python
import numpy as np

# Small sample: Example with 10 individuals
small_sample = np.random.normal(loc=170, scale=10, size=10)  # Heights of 10 men

# Calculate sample mean and standard deviation
small_sample_mean = np.mean(small_sample)
small_sample_std = np.std(small_sample, ddof=1)

small_sample_mean, small_sample_std
```

The output is: (165.8037339228702, 12.348353654252042)

Question No. – 35:

A random sample of size 25 from a population gives the sample standard derivation to be 9.0. Test the hypothesis that the population standard derivation is 10.5. #

Hint (Use chi-square distribution).

Answer:

According to the given problem,

Sample size (n) = 25

Sample Structured Deviation (s) = 9.0

Hypothesized Standard Deviation (σ) = 10.5

The Hypothesis are:

Null Hypothesis (H_0): The population standard deviation $\sigma = 10.5$

Alternative Hypothesis (H_1): The population standard $\sigma \neq 10.5$ (two-tailed test)

We know,

For chi-square ($\chi2$) test statistic:

$$\chi2 = ((n - 1) * s^2)/ \sigma^2$$

where,

(n – 1) → degrees of freedom
(in this case, it is 25 – 1 = 24)
s → Sample Standard Deviation
σ → Population Standard Deviation under the Null Hypothesis

On substituting the values, we get:-

$\chi2 = ((25 - 1) * 9.0^2)/10.5^2 = (24 * 81)/110.25 = 17.63$

Now,

For Critical Values,

For a two-tailed test at the $\alpha = 0.05$ significance level and 24 degrees of freedom, the critical values can be found in the chi-square distribution table.

Therefore,

Lower Critical Value: $\chi2_{0.025,24} = 13.848$

Upper Critical Value: $\chi2_{0.975,24} = 36.415$

For Decision,

If $\chi2$ is less than the lower critical value or greater than the upper critical value, we reject the null hypothesis.

In this case, $\chi2 \approx 17.63$ lies between the critical values 13.848 and 36.415. Therefore, we fail to reject the null hypothesis.

So,

It can be concluded that there is not enough evidence to conclude that the population standard deviation is different from 10.5.

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 37:

100 students of a PW IOI obtained the following grades in Data Science paper: #

Grade :[A, B, C, D, E]

Total Frequency :[15, 17, 30, 22, 16, 100]

Using the χ 2 test , examine the hypothesis that the distribution of grades is uniform.

Answer:

According to the given problem,

Observed frequencies of grades = [15, 17, 30, 22, 16]

Total Number of Students = 100

The Hypothesis are:

Null Hypothesis (H_0): The distribution of grades is uniform.

Alternative Hypothesis (H_1): The distribution of grades is not uniform.

Now,

Expected Frequency = $\frac{Total\ Number\ of\ Students}{Number\ of\ Grades} = \frac{100}{5} = 20$

Therefore,

Expected Frequency = [20, 20, 20, 20, 20]

Again,

For calculation of Chi-Square Test Statistic:

$\chi 2 = \Sigma\ (O_i - E_i)^2/E_i$

where,

O_i → Observed Frequency for each grade

E_i → Expected Frequency for each grade

On calculating the values, we get:-

$\chi 2 = ((15-20)^2/20 + (17-20)^2/20 + (30-20)^2/20 + (22-20)^2/20 + (16-20)^2/20)$

→ $\chi 2 = \frac{25}{20} + \frac{9}{20} + \frac{100}{20} + \frac{4}{20} + \frac{16}{20} = 1.25 + 0.45 + 5 + 0.2 + 0.8 = 7.7$

Similarly,

Degrees of Freedom (df) = Number of categories – 1 = 5 – 1 = 4

Now,

At α = 0.05 and 4 degrees of freedom, the critical value from the chi-square distribution table is approximately 9.488.

For Decision,

If χ2 is greater than the critical value, we reject the null hypothesis.

In this case, χ2 = 7.7, which is less than the critical value of 9.488.

Therefore, we fail to reject the null hypothesis.

So,

It can be concluded that there is not enough evidence to reject the hypothesis that the distribution of grades is uniform.

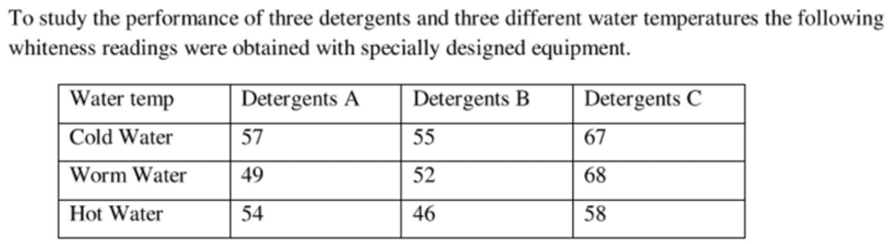
This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 38:

ANOVA Test: #



To study the performance of three detergents and three different water temperatures the following whiteness readings were obtained with specially designed equipment.

Water temp	Detergents A	Detergents B	Detergents C
Cold Water	57	55	67
Worm Water	49	52	68
Hot Water	54	46	58

Answer:

According to the given problem, we have the following information:

Water Temperature	Detergent A	Detergent B	Detergent C
Cold Water	57	55	67
Warm Water	49	52	68
Hot Water	54	46	58

The Hypothesis would be:

Null Hypothesis (H_0): There is no significant difference in the means of the detergents or water temperatures.

Alternative Hypothesis (H_1): At least one detergent or water temperature has a significantly different mean.

We know,

Grand Mean (GM) = $\Sigma X_{ij}/n$

Sum of Squares for Treatments (SST) = $\Sigma^{k}_{I=1} n_i (X'_I - GM)^2$

Sum of Squares for Error (SSE) = $\Sigma^{k}_{I=1} \Sigma^{ni}_{j=1} (X'_{Ij} - X'^{2}_{I})^2$

Total Sum of Squares (TSS) = SST + SSE

Therefore,

In order to calculate ANOVA, the required ANOVA table is as follows:

Source of Variation	Sum of Squares (SS)	Degrees of Freedom (df)	Mean Square (MS)	F – Value
Treatments (Between)	SST	k - 1	SST / (k – 1)	MST / MSE
Error (Within)	SSE	N – k	SSE / (N – k)	
Total	TSS	N - 1		

where,

k → Number of groups

N → Total number of observations

Now,

Calculation:

Step 1: Calculate the Mean of Each Group

Mean of Detergent A = $X'_A = \frac{57+49+54}{3} = \frac{160}{3} = 53.33$

Mean of Detergent B = $X'_B = \frac{55+52+46}{3} = \frac{153}{3} = 51$

Mean of Detergent C = $X'_C = \frac{67+68+58}{3} = \frac{193}{3} = 64.33$

Step 2: Calculate the Grand Mean

$GM = \frac{57+49+54+55+52+46+67+68+58}{9} = \frac{506}{9} = 56.22$

Step 3: Calculate the Sum of Squares

$SST = n_A (X'_A - GM)^2 + n_B (X'_B - GM)^2 + n_C (X'_C - GM)^2$

Here, $n_A = n_B = n_C = 3$

Therefore,

$SST = 3 (53.33 - 56.22)^2 + 3 (51 - 56.22)^2 + 3 (64.33 - 56.22)^2$

→ $SST = 3 (-2.89)^2 + 3 (-5.22)^2 + 3 (8.11)^2$

→ $SST = 3 (8.3521) + 3 (27.2484) + 3 (65.8081)$

→ $SST = 25.0563 + 81.7452 + 197.4243$

→ $SST = 304.2258$

$SSE = (57-53.33)^2 + (49-53.33)^2 + (54-53.33)^2 + (55-51)^2 + (52-51)^2 + (46-51)^2 + (67-64.33)^2 + (68-64.33)^2 + (58-64.33)^2 = 13.44 + 18.75 + 0.44 + 16 + 1 + 25 + 7.11 + 13.44 + 40.11 = 135.29$

$TSS = SST + SSE = 304.2258 + 135.29 = 439.5158$

Step 4: Compute the Mean – Squares

$MST = SST/(k - 1) = 304.2258/2 = 152.113$

$MSE = SSE/(N - k) = 135.29/6 = 22.5483$

Step 5: Calculate F – Value

F – value = MST/MSE = 152.113/22.5483 = 6.745

Step 6: Compare with F – Critical Value

Let us assume,

Significance Level of $\alpha = 0.05$

Degrees of Freedom for Numerator (df_1) = k – 1 = 2

Degrees of Freedom for Denominator (df_2) = N – k = 6

Therefore,

The required Critical F – Value is 5.14

Step 7: Decision

Since the calculated F-value (6.745) is greater than the critical F-value (5.14), we reject the null hypothesis (H_0)

Step 8: Conclusion

There is a significant difference in the means of the detergents or water temperatures. This indicates that the performance of at least one detergent or the effect of one water temperature is significantly different from the others.

This question has also been solved via Python code (marked by a # as they have been solved in both theory as well as coding), in a Jupyter Notebook and it has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Notebook_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Notebook_Codes/Statistics_Codes.ipynb>

Question No. – 39:

How would you create a basic Flask route that displays "Hello, World!" on the homepage?*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Flask_Codes/hello_world.py>

Question No. – 40:

Explain how to set up a Flask application to handle form submissions using POST requests.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/GetPostRequests>

Question No. – 41:

Write a Flask route that accepts a parameter in the URL and displays it on the page.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Flask_Codes/parameter_URL.py>

Question No. – 42:

How can you implement user authentication in a Flask application?*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assi

gnment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/UserAuthentication
>

Question No. – 43:

Describe the process of connecting a Flask app to a SQLite database using SQLAlchemy.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Flask_Codes/sqlalchemy.py>

Question No. – 44:

How would you create a RESTful API endpoint in Flask that returns JSON data?*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/RestfulAPI>

Question No. – 45:

Explain how to use Flask-WTF to create and validate forms in a Flask application.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:***

Question No. – 46:

How can you implement file uploads in a Flask application?*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/FileUploads>

Question No. – 47:

Describe the steps to create a Flask blueprint and why you might use one.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Flask_Codes/blueprint.py>

Question No. – 48:

How would you deploy a Flask application to a production server using Gunicorn and Nginx?*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>

- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/Gunicorn_Nginx>

Question No. – 49:

Make a fully functional web application using flask, MongoDB. Signup, Signin page and after successfully login. Say hello Geeks message at webpage.*

Solution:

This question is a coding question and it has been solved in a ".py" file and has been uploaded to GitHub.

The required Links are:

- ***Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Flask_Codes/WebApplication>

Question No. – 50:

Answer the following questions with respect to Machine Learning:

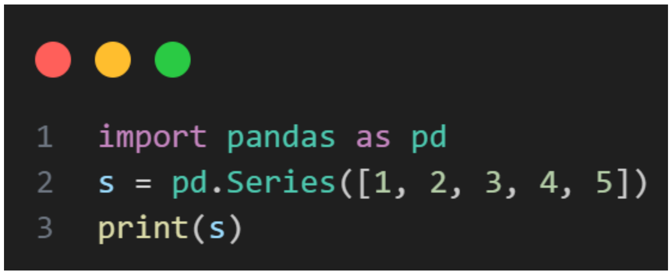
- What is the difference between Series & Dataframes?

Answer:

The difference between Series & Dataframes are as follows:-

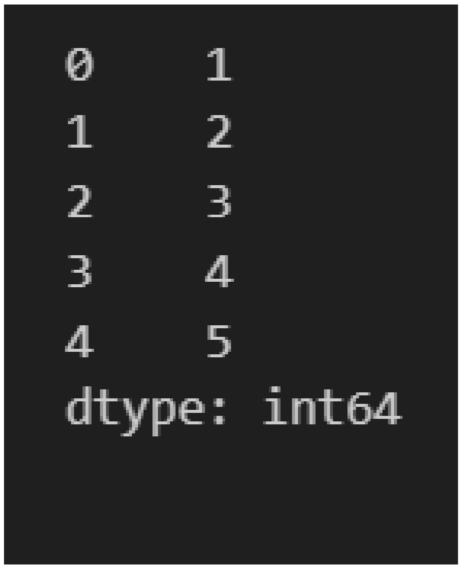
Series	***DataFrames***
A ‘Series’ is a one-dimensional labeled array capable of holding any data type (integer, string, float, etc.).	A ‘DataFrame’ is a two-dimensional labeled data structure with columns of potentially different types.
It is similar to a column in an Excel sheet or a database table.	It is similar to a spreadsheet or a SQL table.
Each element in a Series has an index, which labels the data.	Each column in a DataFrame is a Series.

Example of Series:



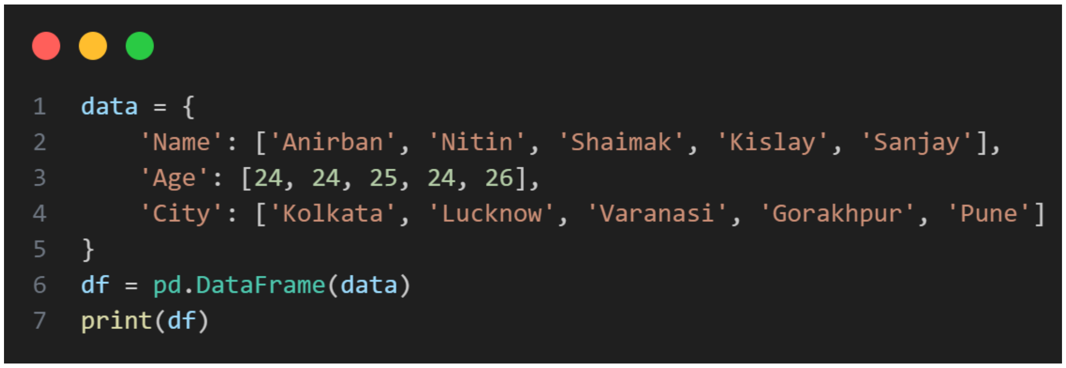
```python
import pandas as pd
s = pd.Series([1, 2, 3, 4, 5])
print(s)
```

Output:



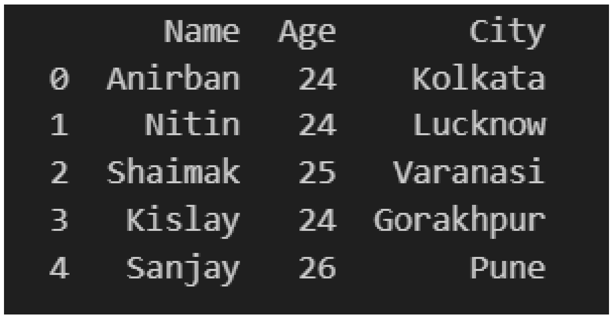
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

Example of DataFrames:



```python
data = {
    'Name': ['Anirban', 'Nitin', 'Shaimak', 'Kislay', 'Sanjay'],
    'Age': [24, 24, 25, 24, 26],
    'City': ['Kolkata', 'Lucknow', 'Varanasi', 'Gorakhpur', 'Pune']
}
df = pd.DataFrame(data)
print(df)
```

Output:

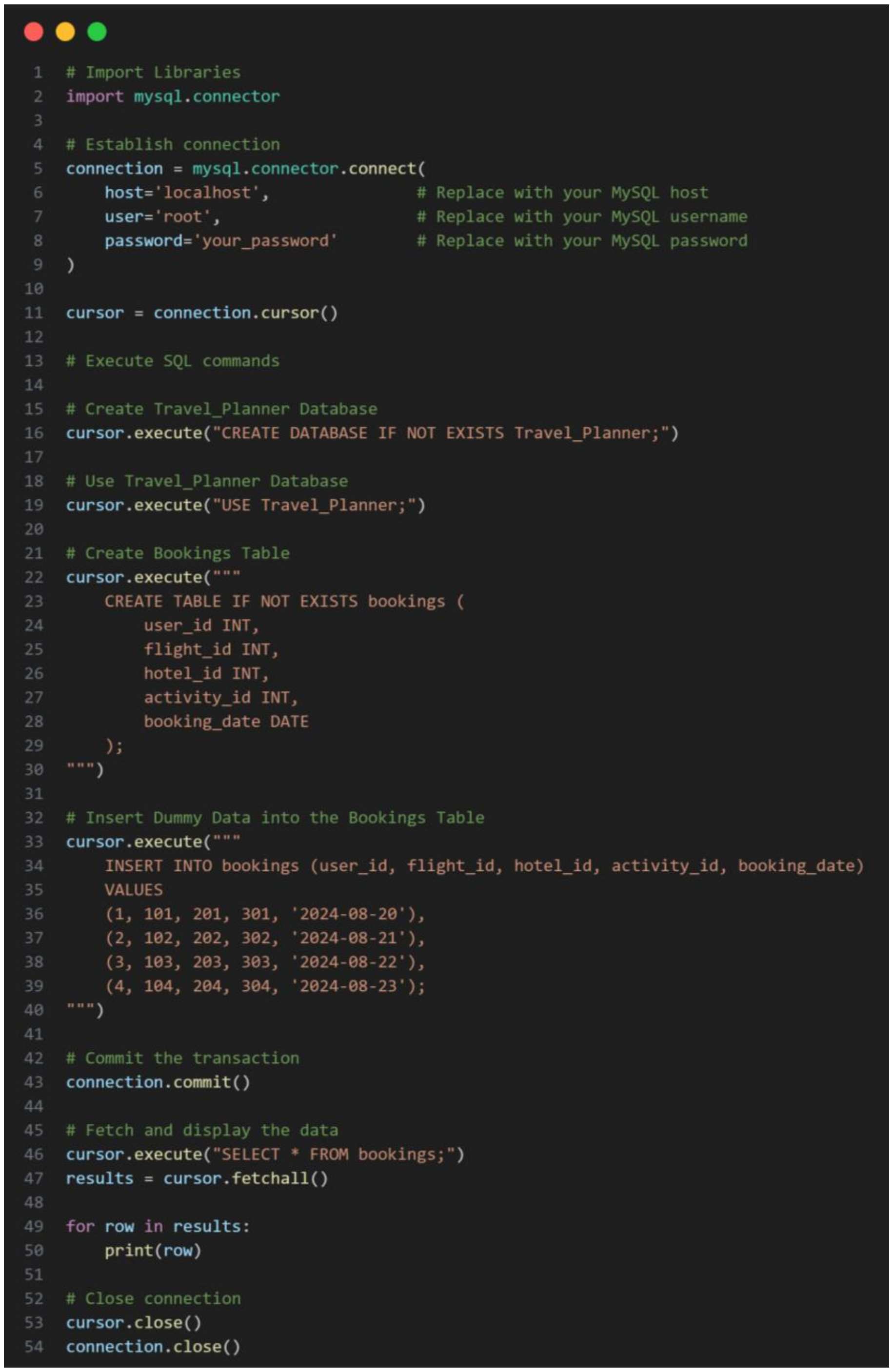


```
      Name  Age       City
0  Anirban   24    Kolkata
1    Nitin   24    Lucknow
2  Shaimak   25   Varanasi
3   Kislay   24  Gorakhpur
4   Sanjay   26       Pune
```

- Create a database name Travel_Planner in mysql, and create a table name bookings in that which having attributes (user_id, INT, flight_id, INT, hotel_id, INT, activity_id, INT, booking_date, DATE). Fill with some dummy value. Now you have to read the content of this table using pandas as dataframe. Show the output.

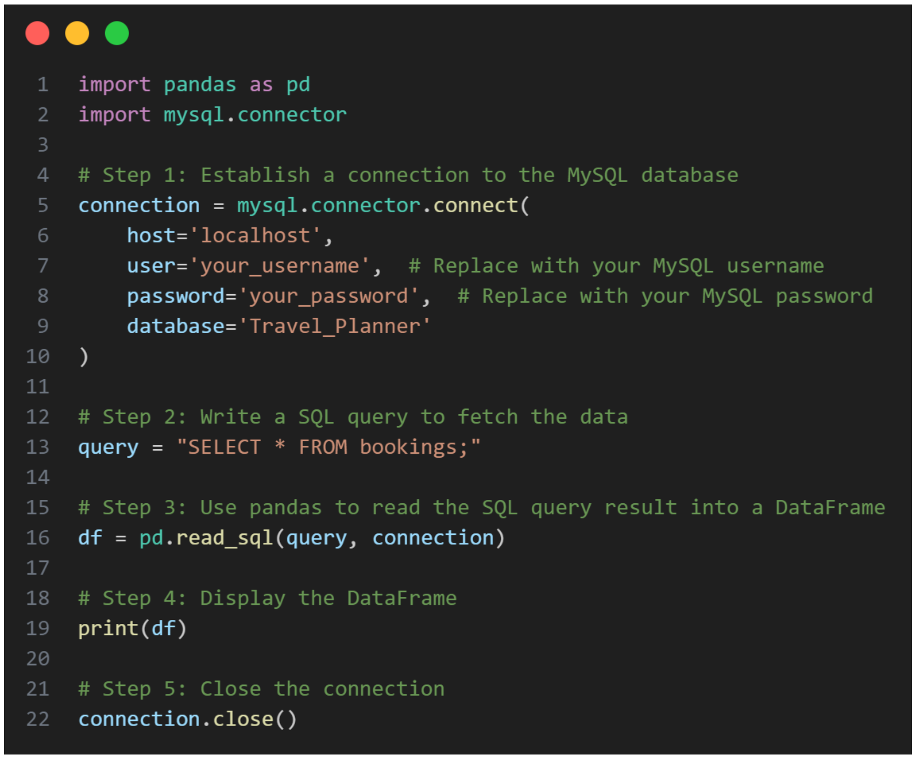
Answer:

➢ ***SQL Query:***



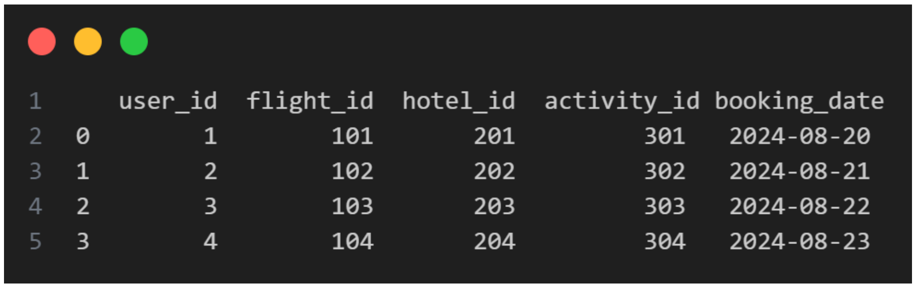
```
# Import Libraries
import mysql.connector

# Establish connection
connection = mysql.connector.connect(
    host='localhost',              # Replace with your MySQL host
    user='root',                   # Replace with your MySQL username
    password='your_password'       # Replace with your MySQL password
)

cursor = connection.cursor()

# Execute SQL commands

# Create Travel_Planner Database
cursor.execute("CREATE DATABASE IF NOT EXISTS Travel_Planner;")

# Use Travel_Planner Database
cursor.execute("USE Travel_Planner;")

# Create Bookings Table
cursor.execute("""
    CREATE TABLE IF NOT EXISTS bookings (
        user_id INT,
        flight_id INT,
        hotel_id INT,
        activity_id INT,
        booking_date DATE
    );
""")

# Insert Dummy Data into the Bookings Table
cursor.execute("""
    INSERT INTO bookings (user_id, flight_id, hotel_id, activity_id, booking_date)
    VALUES
    (1, 101, 201, 301, '2024-08-20'),
    (2, 102, 202, 302, '2024-08-21'),
    (3, 103, 203, 303, '2024-08-22'),
    (4, 104, 204, 304, '2024-08-23');
""")

# Commit the transaction
connection.commit()

# Fetch and display the data
cursor.execute("SELECT * FROM bookings;")
results = cursor.fetchall()

for row in results:
    print(row)

# Close connection
cursor.close()
connection.close()
```

➢ ***Python Implementation:***



```python
import pandas as pd
import mysql.connector

# Step 1: Establish a connection to the MySQL database
connection = mysql.connector.connect(
    host='localhost',
    user='your_username',  # Replace with your MySQL username
    password='your_password',  # Replace with your MySQL password
    database='Travel_Planner'
)

# Step 2: Write a SQL query to fetch the data
query = "SELECT * FROM bookings;"

# Step 3: Use pandas to read the SQL query result into a DataFrame
df = pd.read_sql(query, connection)

# Step 4: Display the DataFrame
print(df)

# Step 5: Close the connection
connection.close()
```

➢ ***Expected Output:***



```
   user_id  flight_id  hotel_id  activity_id booking_date
0        1        101       201          301   2024-08-20
1        2        102       202          302   2024-08-21
2        3        103       203          303   2024-08-22
3        4        104       204          304   2024-08-23
```

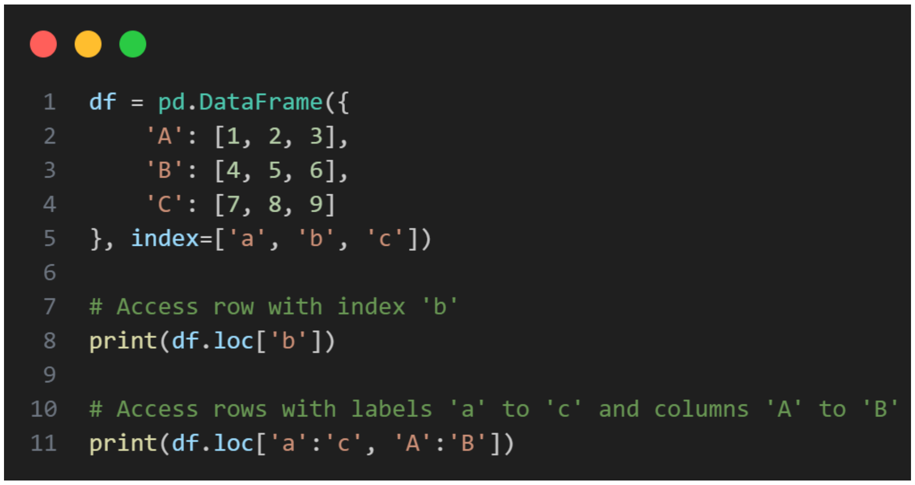
- Difference between loc and iloc.

Answer:

The difference between loc and iloc are as follows:-

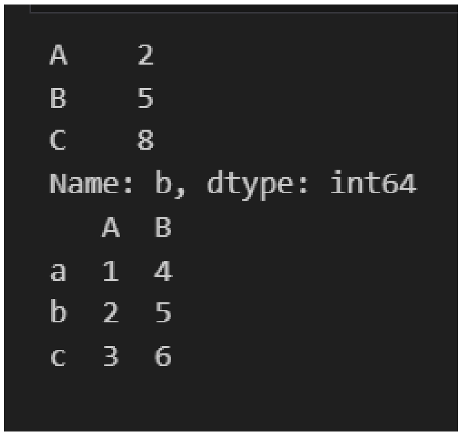
loc	***Iloc***
It refers to label – based indexing.	It refers to integer – based or position – based indexing.
It is used to access a group of rows and columns by labels or a boolean array.	It is used to access a group of rows and columns by integer position(s).
It includes the last element of the range specified.	It does not include the last element of the range specified.

Example of loc:



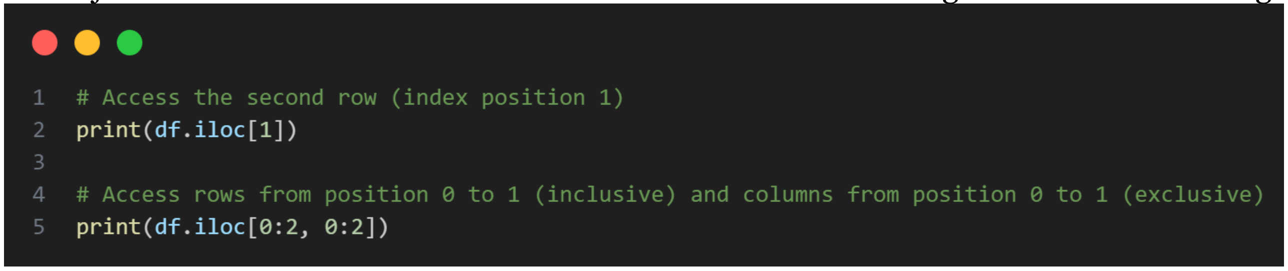
```python
df = pd.DataFrame({
    'A': [1, 2, 3],
    'B': [4, 5, 6],
    'C': [7, 8, 9]
}, index=['a', 'b', 'c'])

# Access row with index 'b'
print(df.loc['b'])

# Access rows with labels 'a' to 'c' and columns 'A' to 'B'
print(df.loc['a':'c', 'A':'B'])
```

Output:



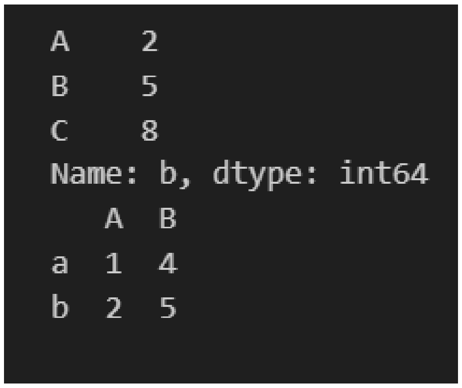
```
A    2
B    5
C    8
Name: b, dtype: int64
   A  B
a  1  4
b  2  5
c  3  6
```

Example of iloc:



```python
# Access the second row (index position 1)
print(df.iloc[1])

# Access rows from position 0 to 1 (inclusive) and columns from position 0 to 1 (exclusive)
print(df.iloc[0:2, 0:2])
```

Output:



```
A    2
B    5
C    8
Name: b, dtype: int64
   A  B
a  1  4
b  2  5
```

- What is the difference between supervised and unsupervised learning?

Answer:

The difference between Supervised and Unsupervised Learning are as follows:-

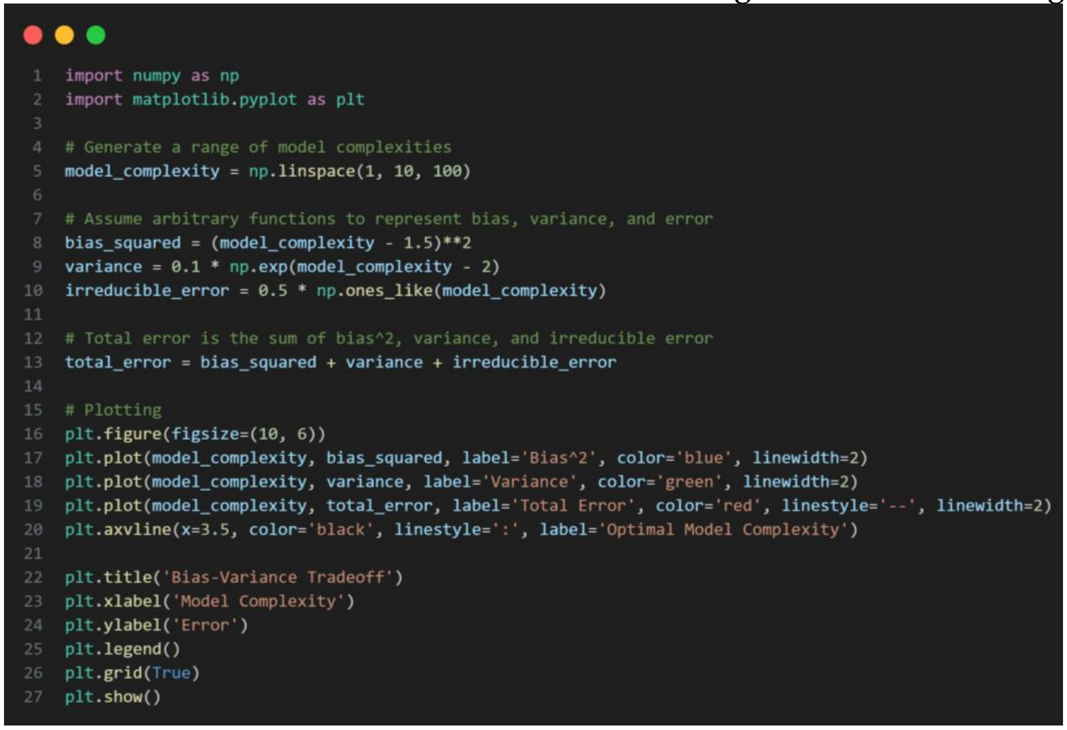
Supervised Learning	***Unsupervised Learning***
In supervised learning, the algorithm is trained on a labeled dataset, which means that each training example is paired with an output label.	In unsupervised learning, the algorithm is trained on an unlabeled dataset, which means that the data has no labels or predefined outcomes.
The goal is to learn a mapping from inputs to outputs, allowing the model to predict the label for new, unseen data.	The goal is to infer the natural structure present within a set of data points.
Common tasks: classification (predicting discrete labels) and regression (predicting continuous values).	Common tasks: clustering (grouping data points), association (finding relationships between variables), dimensionality reduction.
Examples: Linear Regression, Support Vector Machines, Decision Trees, Neural Networks.	*Examples:* K-Means Clustering, Principal Component Analysis (PCA), Hierarchical Clustering, Apriori Algorithm.

- Explain the bias-variance tradeoff.

Answer:

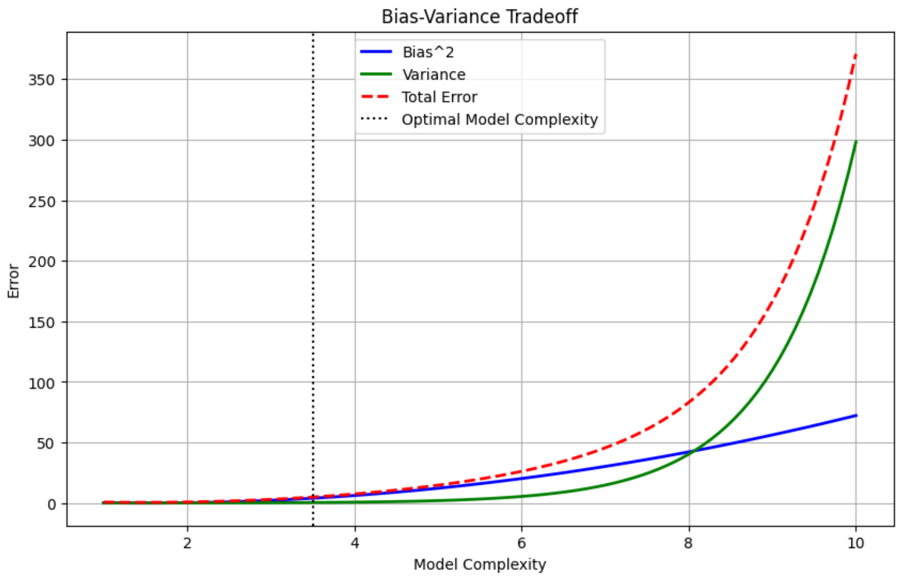
The Bias – Variance Tradeoff is a fundamental concept in machine learning that describes the tradeoff between two sources of error in a predictive model:

- ***Bias:***
  - ✓ Bias is the error introduced by approximating a real-world problem, which may be complex, by a simpler model.
  - ✓ High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- ***Variance:***
  - ✓ Variance is the error introduced by the model's sensitivity to small fluctuations in the training set.
  - ✓ High variance can cause an algorithm to model the noise in the training data rather than the intended outputs (overfitting).
- ***Tradeoff:***
  - ✓ Increasing model complexity typically increases variance and decreases bias.
  - ✓ Decreasing model complexity typically increases bias and decreases variance.
  - ✓ The goal is to find a balance where both bias and variance are minimized, achieving low overall error.
- ***Characteristic Graphical Representation:***
  - ✓ High Bias, Low Variance - Simple models (e.g., linear regression with few features) - Underfitting.
  - ✓ Low Bias, High Variance - Complex models (e.g., deep neural networks) - Overfitting.
  - ✓ Optimal Model - Balanced bias and variance, good generalization on new data.
- ***Sample Example and Output:***
  - ✓ *Python Implementation:*



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

# Generate a range of model complexities
model_complexity = np.linspace(1, 10, 100)

# Assume arbitrary functions to represent bias, variance, and error
bias_squared = (model_complexity - 1.5)**2
variance = 0.1 * np.exp(model_complexity - 2)
irreducible_error = 0.5 * np.ones_like(model_complexity)

# Total error is the sum of bias^2, variance, and irreducible error
total_error = bias_squared + variance + irreducible_error

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(model_complexity, bias_squared, label='Bias^2', color='blue', linewidth=2)
plt.plot(model_complexity, variance, label='Variance', color='green', linewidth=2)
plt.plot(model_complexity, total_error, label='Total Error', color='red', linestyle='--', linewidth=2)
plt.axvline(x=3.5, color='black', linestyle=':', label='Optimal Model Complexity')

plt.title('Bias-Variance Tradeoff')
plt.xlabel('Model Complexity')
plt.ylabel('Error')
plt.legend()
plt.grid(True)
plt.show()
```

✓ *Output:*



- What are precision and recall? How are they different from accuracy?

Answer:

The definition of Precision and Recall and in which way they differ from Accuracy have been explained as follows:-

- ***Precision:***
  - ✓ Precision (also called Positive Predictive Value) is the ratio of correctly predicted positive observations to the total predicted positives.
  - ✓ *Formula:* Precision = TP/(TP + FP)
  - ✓ High precision indicates a low false positive rate.
- ***Recall:***
  - ✓ Recall (also called Sensitivity or True Positive Rate) is the ratio of correctly predicted positive observations to all the observations in the actual class.
  - ✓ *Formula:* Recall = TP/(TP + FN)
  - ✓ High recall indicates a low false negative rate.
- ***Difference From Accuracy:***
  - ✓ Accuracy is the ratio of correctly predicted observations to the total observations.
  - ✓ *Formula:* (TP +TN)/(TP + TN + FP + FN)
  - ✓ Precision and recall focus on the performance of the positive class, while accuracy considers all classes.
- ***Example:*** In a medical test for a disease –
  - ✓ *High Precision:* Few false positives, fewer healthy people are misdiagnosed as sick.
  - ✓ *High Recall:* Few false negatives, most sick people are correctly identified as sick.
  - ✓ *High Accuracy:* Overall correct diagnoses, but could be misleading if the class distribution is imbalanced (e.g., very few cases of the disease).

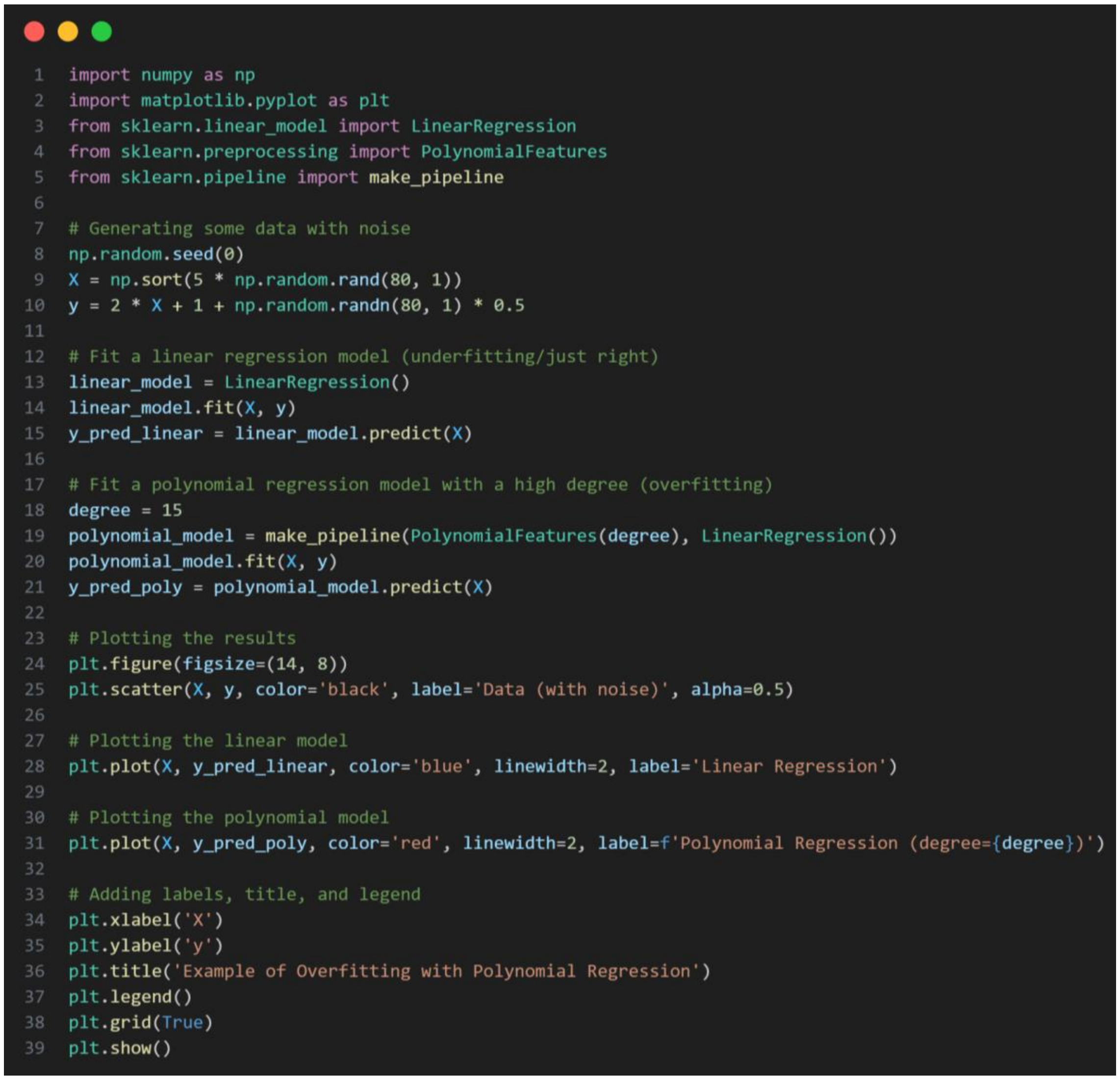
- What is overfitting and how can it be prevented?

Answer:

Overfitting is a phenomenon which leads a Machine Learning Model to capture noise and details in the training data to an extent that it negatively impacts the performance of the model on new data. It happens when the model is too complex, with too many parameters relative to the number of observations.

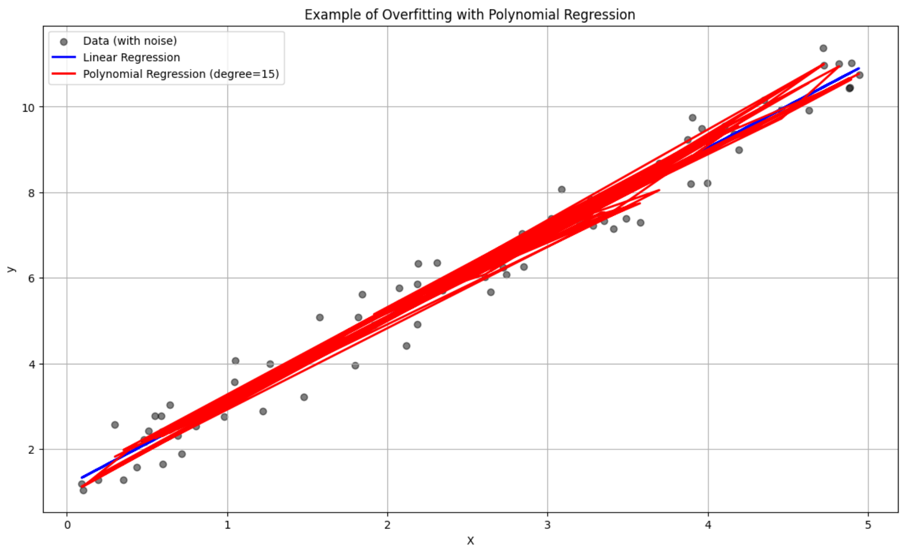
Example of Overfitting: Consider an example of usage of Polynomial Regression to fit data points, where some data will be generated with a simple linear relationship and then fit both a simple linear model and an overly complex polynomial model to this data.

Python Implementation:



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

# Generating some data with noise
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1))
y = 2 * X + 1 + np.random.randn(80, 1) * 0.5

# Fit a linear regression model (underfitting/just right)
linear_model = LinearRegression()
linear_model.fit(X, y)
y_pred_linear = linear_model.predict(X)

# Fit a polynomial regression model with a high degree (overfitting)
degree = 15
polynomial_model = make_pipeline(PolynomialFeatures(degree), LinearRegression())
polynomial_model.fit(X, y)
y_pred_poly = polynomial_model.predict(X)

# Plotting the results
plt.figure(figsize=(14, 8))
plt.scatter(X, y, color='black', label='Data (with noise)', alpha=0.5)

# Plotting the linear model
plt.plot(X, y_pred_linear, color='blue', linewidth=2, label='Linear Regression')

# Plotting the polynomial model
plt.plot(X, y_pred_poly, color='red', linewidth=2, label=f'Polynomial Regression (degree={degree})')

# Adding labels, title, and legend
plt.xlabel('X')
plt.ylabel('y')
plt.title('Example of Overfitting with Polynomial Regression')
plt.legend()
plt.grid(True)
plt.show()
```

Output:



Some of the prevention techniques for overfitting are as follows:-

- ***Cross-Validation:*** Use techniques like k-fold cross-validation to ensure the model generalizes well to unseen data.
- ***Model Simplification:*** Reduce the complexity of the model by selecting fewer features or using a simpler algorithm.
- ***Regularization:*** Add a penalty for larger coefficients to the loss function (e.g., L1 or L2 regularization).
- ***Early Stopping:*** Stop training the model when the performance on the validation set starts to degrade.
- ***Data Augmentation:*** Increase the size and variability of the training dataset through techniques like rotation, flipping, scaling, etc.
- ***Ensemble Methods:*** Combine predictions from multiple models to reduce overfitting (e.g., Random Forests, Boosting).
- ***Pruning (in Decision Tree):*** Remove parts of the tree that do not provide power in predicting target variables.
- ***Dropout (in Neural Networks):*** Randomly drop units (along with their connections) from the neural network during training to prevent co-adaptation.

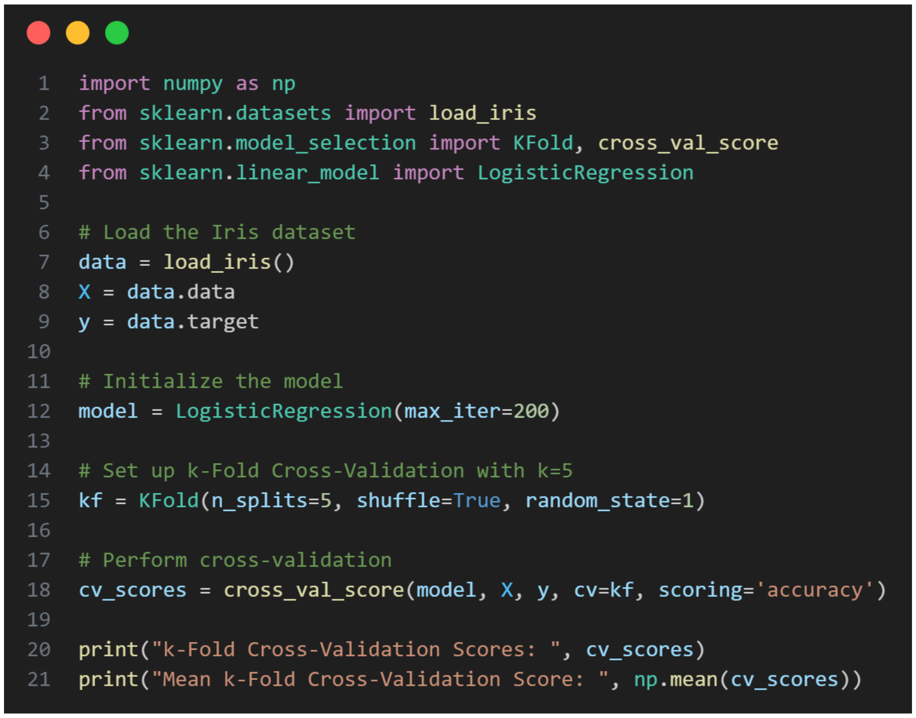
- Explain the concept of cross-validation.

Answer:

Cross-validation is a statistical method used to evaluate the generalization performance of a machine learning model. It helps to ensure that the model's performance is not just good on the training data but also on unseen data. The primary goal is to assess how the model will perform in practice.

Some common types of Cross-validation are as follows:-

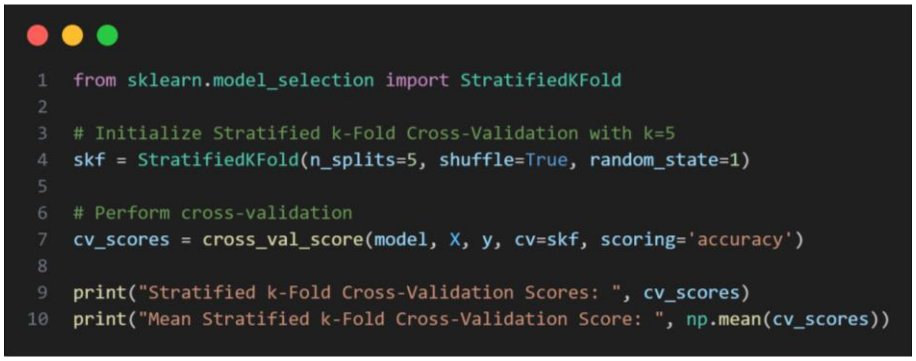
➢ ***K-Fold Cross-Validation:***
  - ✓ The dataset is randomly partitioned into 'k' equal-sized folds (subsets).
  - ✓ The model is trained 'k' times, each time using 'k-1' folds for training and the remaining fold for validation.
  - ✓ The process is repeated 'k' times (iterations), each time with a different fold as the validation set.
  - ✓ The performance metric (e.g., accuracy, RMSE) is averaged over all 'k' iterations to get an overall performance estimate.
  - ✓ *Example (Python Implementation):*



```python
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression

# Load the Iris dataset
data = load_iris()
X = data.data
y = data.target

# Initialize the model
model = LogisticRegression(max_iter=200)

# Set up k-Fold Cross-Validation with k=5
kf = KFold(n_splits=5, shuffle=True, random_state=1)

# Perform cross-validation
cv_scores = cross_val_score(model, X, y, cv=kf, scoring='accuracy')

print("k-Fold Cross-Validation Scores: ", cv_scores)
print("Mean k-Fold Cross-Validation Score: ", np.mean(cv_scores))
```

➢ ***Stratified k-Fold Cross-Validation:***
  - ✓ Similar to k-fold cross-validation, but the folds are stratified so that they contain approximately the same proportion of class labels as the original dataset.

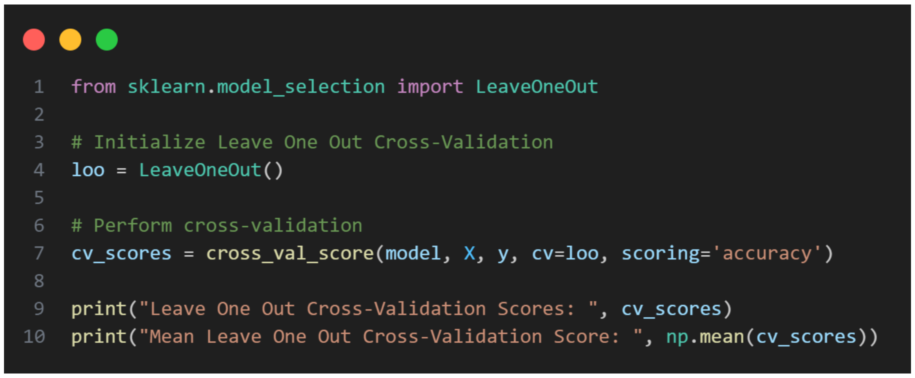
- ✓ Commonly used for imbalanced datasets.
- ✓ *Example (Python Implementation):*



```python
from sklearn.model_selection import StratifiedKFold

# Initialize Stratified k-Fold Cross-Validation with k=5
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)

# Perform cross-validation
cv_scores = cross_val_score(model, X, y, cv=skf, scoring='accuracy')

print("Stratified k-Fold Cross-Validation Scores: ", cv_scores)
print("Mean Stratified k-Fold Cross-Validation Score: ", np.mean(cv_scores))
```

➢ ***Leave-One-Out-Cross-Validation:***

- ✓ A special case of k-fold cross-validation where 'k' is equal to the number of data points.
- ✓ Each data point is used once as the validation set, and the remaining points are used for training.
- ✓ *Example (Python Implementation):**i***



```python
from sklearn.model_selection import LeaveOneOut

# Initialize Leave One Out Cross-Validation
loo = LeaveOneOut()

# Perform cross-validation
cv_scores = cross_val_score(model, X, y, cv=loo, scoring='accuracy')

print("Leave One Out Cross-Validation Scores: ", cv_scores)
print("Mean Leave One Out Cross-Validation Score: ", np.mean(cv_scores))
```

The Benefits of Cross-Validation can be explained as follows:-

➢ ***More Reliable Evaluation:*** Instead of relying on a single train-test split, cross-validation provides a more robust assessment by using different subsets of the data for training and testing.

➢ ***Better Generalization:*** It helps to avoid overfitting by ensuring that the model is tested on different parts of the data.

➢ ***Model Selection:*** Cross-validation is often used in conjunction with hyperparameter tuning to select the best model configuration.

- What is the difference between a classification and a regression problem?

Answer:

The difference between a Classification and a Regression problem are as follows:-

Classification	***Regression***
Classification involves predicting a discrete label or category for a given input.	Regression involves predicting a continuous value for a given input.
The output variable is categorical.	The output variable is continuous.
Common algorithms: Logistic Regression, Decision Trees, Random Forest, Support Vector Machines, Neural Networks.	*Common algorithms:* Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression, Neural Networks.
Examples: Spam detection (spam/not spam), image recognition (cat/dog/other), medical diagnosis (disease present/absent).	*Examples:* Predicting house prices, forecasting stock prices, estimating temperature.

- Explain the concept of ensemble learning.

Answer:

Ensemble Learning is a technique that combines the predictions of multiple machine learning models to improve the overall performance and robustness of the model. The idea is that a group of weak learners can come together to form a strong learner. Ensemble methods are particularly effective in reducing overfitting and improving predictive accuracy.

The types of Ensemble Learning include the following:-

- ***Bagging (Bootstrap Aggregating):***
  - ✓ Multiple instances of a model are trained on different subsets of the training data, created by sampling with replacement.
  - ✓ The final prediction is made by averaging (regression) or majority voting (classification) the predictions of all models.
  - ✓ *Example*: Random Forest, where multiple decision trees are trained on different bootstrapped samples of the data.
- ***Boosting:***
  - ✓ Models are trained sequentially, each new model focusing on correcting the errors of the previous models.

  - ✓ Each model in the sequence is trained on a weighted version of the data, where misclassified instances from the previous model are given higher weights.
  - ✓ *Example*: AdaBoost, Gradient Boosting, XGBoost.

- ***Voting:***
  - ✓ Multiple models are trained, and their predictions are combined using a voting mechanism.
  - ✓ For classification, the final prediction is determined by majority vote (hard voting) or by averaging the predicted probabilities (soft voting).
  - ✓ For regression, the final prediction is the average of all model predictions.

- What is gradient descent and how does it work?

Answer:

Gradient Descent is an optimization algorithm used to minimize the cost (loss) function of a model. It iteratively adjusts the model's parameters to find the values that minimize the cost function, thereby improving the model's performance.

The working mechanism of Gradient Descent has been explained as follows:-

- ***Initialization:*** Start with an initial set of parameters (weights) often initialized randomly.
- ***Compute Gradient:*** Calculate the gradient of the cost function with respect to each parameter. The gradient is a vector of partial derivatives indicating the direction and rate of the steepest increase of the cost function.
- ***Update Parameters:*** Adjust the parameters in the opposite direction of the gradient to move towards the minimum of the cost function. The amount of adjustment is controlled by the learning rate (α).

$$\theta := \theta - \alpha \nabla J(\theta)$$

where,

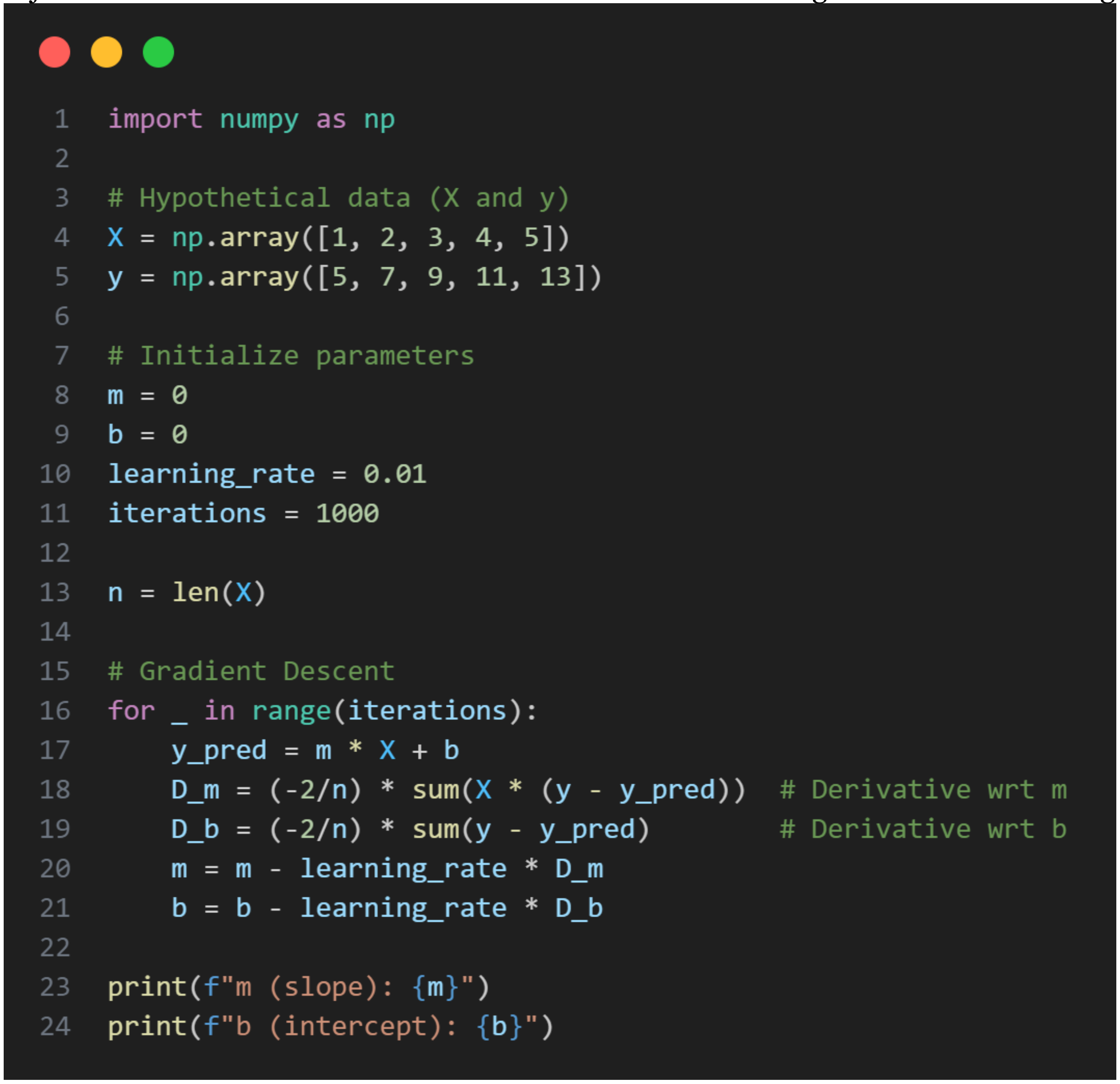
$\theta \rightarrow$ model parameters

$\alpha \rightarrow$ learning rate

$\nabla J(\theta) \rightarrow$ gradient of cost function J with respect to θ

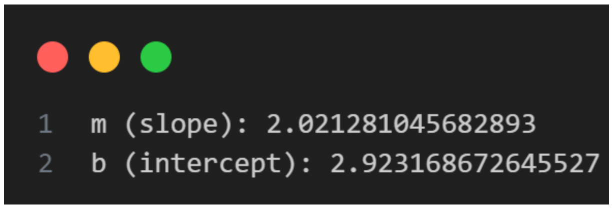
- ***Iterate:*** Repeat the process of computing the gradient and updating the parameters until convergence is achieved (i.e., the change in the cost function is below a certain threshold or a specified number of iterations is reached).

Example (Python Implementation):



```python
import numpy as np

# Hypothetical data (X and y)
X = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])

# Initialize parameters
m = 0
b = 0
learning_rate = 0.01
iterations = 1000

n = len(X)

# Gradient Descent
for _ in range(iterations):
    y_pred = m * X + b
    D_m = (-2/n) * sum(X * (y - y_pred))  # Derivative wrt m
    D_b = (-2/n) * sum(y - y_pred)        # Derivative wrt b
    m = m - learning_rate * D_m
    b = b - learning_rate * D_b

print(f"m (slope): {m}")
print(f"b (intercept): {b}")
```

Output:



```
m (slope): 2.021281045682893
b (intercept): 2.923168672645527
```

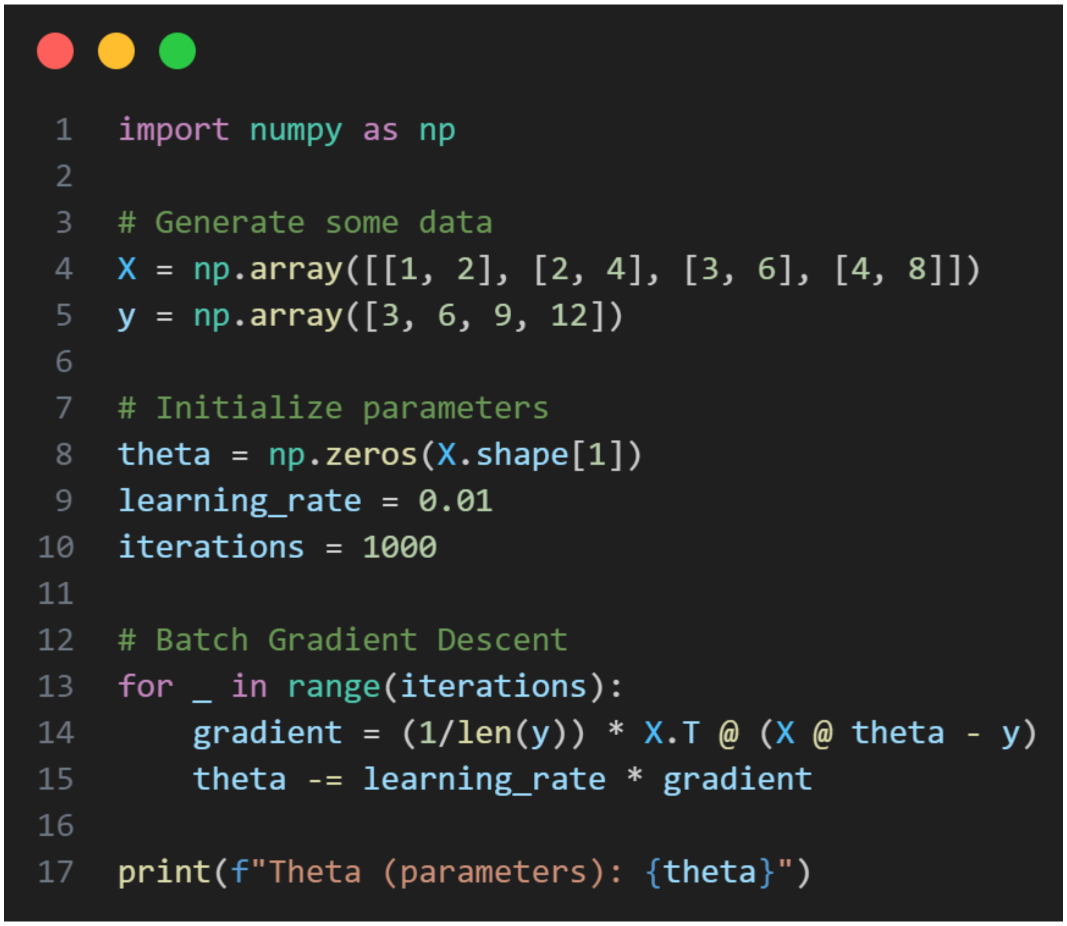
- Describe the difference between batch gradient descent and stochastic gradient descent.

Answer:

The difference between Batch Gradient Descent and Stochastic Gradient Descent are as follows:-

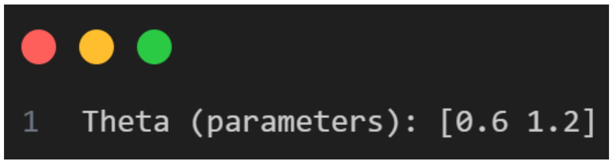
Batch Gradient Descent	***Stochastic Gradient Descent***
Overview: Uses the entire training dataset to compute the gradient of the cost function for each iteration.	*Overview:* Uses a single training example to compute the gradient of the cost function for each iteration.
The gradient of the cost function is computed, using all training examples and the parameters are updated, based on the computed gradient.	The training data is shuffled randomly and for each training example, the gradient of the cost function is computed and the parameters are updated, based on the computed gradient.
Convergence is more stable and smooth.	Convergence is less stable and can be noisy.
Guarantees finding the global minimum for convex functions.	Might need more iterations to converge to the global minimum and can potentially escape local minima due to its noisier updates.
Computationally expensive and slow for large datasets.	Faster iterations and updates, making it suitable for large datasets.
Requires significant memory to store the entire dataset.	Requires less memory since it processes one example at a time.
Use Case: Suitable for small to medium-sized datasets where computational resources are not a constraint.	*Use Case:* Suitable for large datasets and online learning scenarios where data arrives in a stream.

Example of Batch Gradient Descent (Python Implementation):



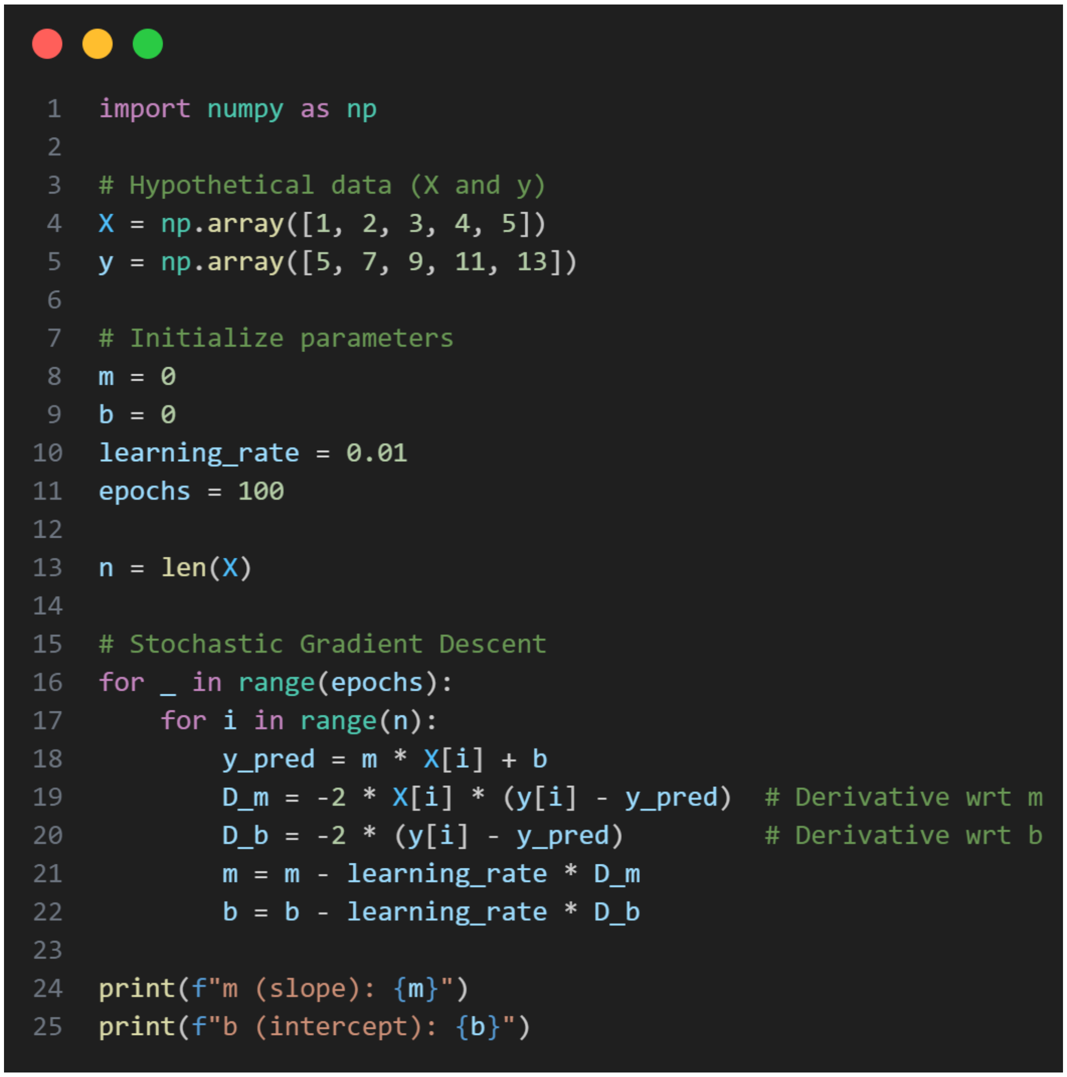
```python
import numpy as np

# Generate some data
X = np.array([[1, 2], [2, 4], [3, 6], [4, 8]])
y = np.array([3, 6, 9, 12])

# Initialize parameters
theta = np.zeros(X.shape[1])
learning_rate = 0.01
iterations = 1000

# Batch Gradient Descent
for _ in range(iterations):
    gradient = (1/len(y)) * X.T @ (X @ theta - y)
    theta -= learning_rate * gradient

print(f"Theta (parameters): {theta}")
```

Output:



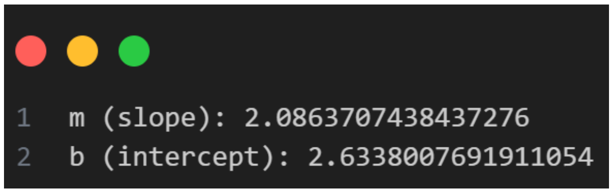
```
Theta (parameters): [0.6 1.2]
```

Example of Stochastic Gradient Descent (Python Implementation):



```python
import numpy as np

# Hypothetical data (X and y)
X = np.array([1, 2, 3, 4, 5])
y = np.array([5, 7, 9, 11, 13])

# Initialize parameters
m = 0
b = 0
learning_rate = 0.01
epochs = 100

n = len(X)

# Stochastic Gradient Descent
for _ in range(epochs):
    for i in range(n):
        y_pred = m * X[i] + b
        D_m = -2 * X[i] * (y[i] - y_pred)  # Derivative wrt m
        D_b = -2 * (y[i] - y_pred)         # Derivative wrt b
        m = m - learning_rate * D_m
        b = b - learning_rate * D_b

print(f"m (slope): {m}")
print(f"b (intercept): {b}")
```

Output



```
m (slope): 2.0863707438437276
b (intercept): 2.6338007691911054
```

- What is the curse of dimensionality in machine learning?

Answer:

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings. As the number of features (dimensions) increases, the volume of the space increases exponentially, making the available data sparse. This sparsity is problematic because it:

- Increases computational cost and memory requirements.
- Makes it difficult to visualize and interpret data.
- Causes overfitting, as models may fit the noise in high-dimensional data.
- Increases the risk that distances between points become less meaningful, affecting the performance of distance-based algorithms like k-nearest neighbors and clustering.

- Explain the difference between L1 and L2 regularization.

Answer:

The difference between L1 and L2 Regularization are as follows:-

L1 Regularization	***L2 Regularization***
Adds the absolute value of the magnitude of coefficients as a penalty term to the loss function.	Adds the squared magnitude of coefficients as a penalty term to the loss function.
Encourages sparsity, effectively performing feature selection by driving some coefficients to zero.	Encourages small, evenly distributed coefficients, reducing the model's complexity without eliminating features.
Penalty Term: $\lambda \Sigma^{n}_{i=1}\|w_i\|$	*Penalty Term:* $\lambda \Sigma^{n}_{i=1} w_i^2$

- What is a confusion matrix and how is it used?

Answer:

A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the prediction results by comparing actual target values with those predicted by the model.

The structure of a Confusion Matrix can be summarized as follows:-

- ***True Positives (TP):*** Correctly predicted positive instances.
- ***True Negatives (TN):*** Correctly predicted negative instances.
- ***False Positives (FP):*** Incorrectly predicted positive instances (Type I error).
- ***False Negatives (FN):*** Incorrectly predicted negative instances (Type II error).

The structure can be visualized by the following example (sample):

	Predicted Positive	Predicted Negative
Actual Positive	TP	TN
Actual Negative	FP	FN

Usage of Confusion Matrix are:-

- ***Accuracy:*** (TP + TN)/(TP + TN + FP + FN)
- ***Precision:*** TP/(TP + FP)
- ***Recall (Sensitivity):*** TP/(TP + FN)
- ***F1 – Score:*** Harmonic mean of precision and recall →
  2 * (Precision * Recall)/(Precision + Recall)

- Define AUC-ROC curve.

Answer:

AUC-ROC Curve stands for "Area Under the Receiver Operating Characteristic Curve." It is a performance measurement for classification models at various threshold settings.

Its components are:-

- ***ROC:*** A graph showing the performance of a classification model by plotting the true positive rate (recall) against the false positive rate at different threshold levels.
- ***AUC:*** The area under the ROC curve, which provides a single measure of overall model performance. It ranges from 0 to 1, with 1 indicating perfect performance and 0.5 indicating random guessing.

- Explain the k-nearest neighbors algorithm.

Answer:

k-Nearest Neighbors (k-NN) is a simple, instance-based learning algorithm used for both classification and regression tasks.

The working mechanism of k-NN can be explained as follows:-

- ***Choose 'k':*** Select the number of nearest neighbors 'k' to consider.
- ***Compute Distance:*** Calculate the distance between the input sample and all training samples using a distance metric (e.g., Euclidean distance).
- ***Find Nearest Neighbors:*** Identify the k training samples that are closest to the input sample.
- ***Predict:***
  - ✓ *Classification:* Assign the most frequent class label among the k nearest neighbors to the input sample (majority vote).
  - ✓ *Regression:* Compute the average (or weighted average) of the k nearest neighbors' values to make the prediction.

Example of k-Nearest Neighbor:

If k=3 and the nearest neighbors of a test point belong to classes [0, 1, 1], then the predicted class would be 1 (majority vote).

Advantages and Disadvantages:

The advantages of k-nearest neighbor algorithm are as follows:-

- Simple and easy to implement.
- No training phase (lazy learning), making it suitable for small datasets.

The disadvantages of k-nearest neighbor algorithm are as follows:-

- Computationally expensive and slow for large datasets.
- Sensitive to the scale of data, requiring normalization or standardization of features and performance can be affected by the choice of k and the distance metric.

- Explain the basic concept of a Support Vector Machine (SVM).

Answer:

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. The main idea of SVM is to find the optimal hyperplane that best separates the data points of different classes in the feature space.

The key concepts of SVM are as follows:-

- ***Hyperplane:*** A decision boundary that separates data points of different classes. In a 2D space, it's a line; in a 3D space, it's a plane; and in higher dimensions, it's a hyperplane.
- ***Support Vectors:*** Data points that are closest to the hyperplane. These points are critical in defining the position and orientation of the hyperplane.
- ***Margin:*** The distance between the hyperplane and the nearest data points from both classes. SVM aims to maximize this margin to ensure better generalization.

- How Does the Kernel Trick Work in SVM?

Answer:

The kernel trick allows SVM to handle non-linearly separable data by transforming the original feature space into a higher-dimensional space where the data becomes linearly separable and it works in the following ways:-

- ***Transformation:*** Data is mapped to a higher-dimensional space using a kernel function.
- ***Linear Separation:*** In this new space, a linear hyperplane can be used to separate the classes.
- ***Kernel Function:*** Instead of explicitly transforming the data, the kernel function computes the dot product in the higher-dimensional space directly, which is computationally efficient.

- What are the different types of kernels used in SVM and when would you use each?

Answer:

The different types of kernels used in SVM along with their use cases have been represented in a tabular form as follows:-

Common Kernel Functions	***Function***	***Use Case***
Linear Kernel	$K(x,x') = x*x'$	When the data is linearly separable or when the number of features is very large relative to the number of samples.
Polynomial Kernel	$K(x,x') = (x*x' + c)^d$	When the data is not linearly separable and there is an interaction between features. The degree d and constant c can be tuned.
Radial Basis Function (RBF) or Gaussian Kernel	$K(x,x')=\exp(-\gamma\|x-x'\|2)$	When the decision boundary is very complex. The parameter γ controls the width of the Gaussian.
Sigmoid Kernel	$K(x,x') = \tanh(\alpha x\cdot x'+c)$	Used in neural networks. It behaves like a two-layer, perceptron model.

- What is the hyperplane in SVM and how is it determined?

Answer:

The hyperplane is the decision boundary that separates different classes in the feature space.

It is determined as follows:-

- SVM finds the hyperplane by maximizing the margin, which is the distance between the hyperplane and the nearest data points (support vectors) from each class.
- The optimal hyperplane is determined by solving the following optimization problem:

$$\frac{min}{w,b}\frac{1}{2}||w||^2$$

subject to the constraints:

$$y_i(w\cdot x_i+b)\geq 1 \qquad \forall i$$

where, w → weight vector; b → bias term; x_I → feature vectors; y_i → class labels.

- What are the pros and cons of using a Support Vector Machine (SVM)?

Answer:

Pros:- The pros of a Support Vector Machine (SVM) are as follows:-

- ***Effective in High Dimension:*** Works well with high-dimensional data and when the number of dimensions exceeds the number of samples.
- ***Memory Efficient:*** Uses a subset of training points (support vectors) in the decision function, making it memory efficient.
- ***Versatile:*** Can be used for both linear and non-linear classification using appropriate kernel functions.
- ***Robust to Overfitting:*** Particularly when using a soft margin, it is less prone to overfitting.

Cons:- The cons of a Support Vector Machine (SVM) are follows:-

- ***Computationally Intensive:*** Training can be slow and computationally intensive, especially for large datasets.
- ***Choice of Kernel:*** Requires careful selection of the kernel function and tuning of hyperparameters, which can be complex.
- ***Less Interpretability:*** The resulting model, especially with non-linear kernels, can be less interpretable compared to simpler models like logistic regression.

- Explain the difference between a hard margin and a soft margin SVM.

Answer:

The difference between Hard Margin and Soft Margin SVM are as follows:-

Parameters	***Hard Margin SVM***	***Soft Margin SVM***
Assumption	Assumes that the data is perfectly linearly separable.	Allows for some misclassification or data points to lie within the margin.
Constraint	No data points can lie within the margin or be misclassified.	Introduces slack variables to handle misclassification and margin violations.
Objective	Maximizes the margin without allowing any misclassification.	Maximizes the margin while minimizing the classification error, controlled by a regularization parameter C.
Use Case	Suitable when data is clean and separable without errors.	Suitable for real-world data where there may be noise and overlap between classes. It balances the trade-off between maximizing the margin and minimizing the classification error.

- Describe the process of constructing a decision tree

Answer:

The process of constructing a Decision Tree can be explained as follows:-

- ***Select the Best Feature to Split:*** Use a criterion like information gain or Gini impurity to select the feature that best separates the data.
- ***Split the Data:*** Divide the dataset into subsets based on the selected feature.
- ***Repeat the Process:*** Recursively apply the above steps to each subset, treating each as a new dataset.
- ***Stopping Conditions:*** Stop when one of the following conditions are not met –
  - ✓ All instances in a node belong to the same class.
  - ✓ No remaining features to split on.
  - ✓ A pre-defined maximum tree depth is reached.
  - ✓ The number of instances in a node is below a pre-defined threshold.

- Describe the working principle of a decision tree.

Answer:

A Decision Tree operates by starting at the root node, where the entire dataset is considered. At each node, it evaluates possible splits based on a chosen criterion, such as information gain or Gini impurity. The algorithm selects the split that optimizes this criterion, typically by maximizing the information gain or minimizing the Gini impurity. The dataset is then divided into subsets according to the selected feature and split. This process is recursively applied to each resulting child node, continuously dividing the data and creating sub-nodes. The tree-building process continues until a stopping condition is met, such as a maximum depth or minimum node size. At this point, the algorithm assigns the most common class label within each node to the corresponding leaf node, completing the decision tree.

- What is information gain and how is it used in decision trees?

Answer:

Information Gain measures the reduction in entropy (uncertainty) after a dataset is split on a feature. It helps in selecting the feature that provides the most significant reduction in uncertainty.

It is represented as:

$$\text{Information Gain (D, A)} = \text{Entropy(D)} - \Sigma_{v \in \text{Values(A)}} \frac{|Dv|}{|D|} \text{Entropy}(D_v)$$

where,

D → Entropy of the Dataset (D)

A → Feature being considered

D_v → Subset of D where feature A has value v

Usage of information gain in Decision Trees:-

- At each node, calculate the information gain for each feature.
- Select the feature with the highest information gain to split the data.
- The process continues recursively for each subset.

- Explain Gini impurity and its role in decision trees.

Answer:

Gini Impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

It is represented as:

$$\text{Gini (D)} = 1 - \Sigma^{C}_{i=1}(pi)^2$$

where,

C → Number of classes

p_i → proportion of instances in class i in the dataset D.

Role of Gini impurity in Decision Trees:-

- Gini impurity is used as a criterion to evaluate splits.
- At each node, calculate the Gini impurity for each feature.
- Select the feature that results in the lowest Gini impurity after the split.
- This process helps in creating nodes that are as pure as possible (i.e., nodes with instances of mostly one class).

- What are the advantages and disadvantages of decision trees?

Answer:

Advantages:- The advantages of Decision Trees are as follows:-

- ***Easy to Understand and Interpret:*** The model can be visualized, and the rules are easy to understand.
- ***Requires Little Data Preparation:*** No need for normalization, scaling, or imputation of missing values.
- ***Handles Both Numerical and Categorical Data:*** Versatile in handling different types of data.
- ***Non – Parametric:*** No assumptions about the distribution of data.
- ***Feature Importance:*** Can identify important features in the dataset.

Disadvantages:- The disadvantages of Decision Trees are as follows:-

- ***Overfitting:-*** Decision trees can easily overfit, especially with complex trees.
- ***Unstable:-*** Small changes in data can result in a completely different tree.
- ***Bias:-*** Greedy algorithms used for splitting may not always produce the optimal tree.
- ***Complexity:-*** Prone to becoming complex with large datasets, leading to difficulties in interpretation.
- ***Performance:-*** Often less accurate than other advanced methods like ensemble methods (e.g., Random Forests, Gradient Boosting).

- How do random forests improve upon decision trees?

Answer:

Random Forests improve upon Decision Trees in several ways, enhancing their predictive performance and robustness. A detailed comparison for the same is as follows:-

Parameters	***Decision Trees***	***Random Forests***
Overfitting Reduction	Single decision trees can easily overfit the training data, especially if they are very deep. This means they may perform well on the training data but poorly on unseen data.	By averaging the predictions of multiple decision trees (ensemble approach), random forests reduce overfitting. The variability of individual trees is averaged out, leading to better generalization on unseen data.

Variance Reduction	The predictions of a single decision tree can be highly variable because a small change in the data can lead to a completely different tree structure.	The use of multiple trees, each trained on different subsets of the data and features, reduces this variance. The ensemble of trees provides a more stable and robust prediction.
Feature Randomization	Decision trees use all features when making splits, which can lead to models that are overly reliant on specific features or relationships in the training data.	At each split, random forests use only a random subset of features. This feature randomization introduces diversity among the trees and helps in capturing different patterns in the data. It also reduces the risk of the model becoming overly dependent on any single feature.
Bias – Variance Tradeoff	Single decision trees may have high variance and low bias, resulting in overfitting.	By combining multiple trees, random forests achieve a balance between bias and variance. While individual trees may have high variance, the averaging of multiple trees helps in reducing variance while maintaining a low bias.
Training and Testing Stability	A single decision tree's performance can be sensitive to the specific training data, leading to variability in predictions.	The ensemble of trees in a random forest provides more stable and consistent performance. The averaging of predictions across multiple trees smooths out individual anomalies.
Handling Missing Values	Basic decision trees may struggle with missing values, requiring additional techniques for imputation or handling.	Random forests can handle missing values more effectively. They can use surrogate splits (alternative splits) to handle missing data, and the averaging of predictions across trees helps mitigate the impact of missing values.
Feature Importance	Decision trees can provide insights into feature importance based on their structure.	Random forests provide a more reliable measure of feature importance by averaging the importance scores across all trees, which gives a more robust indication of feature relevance.

Model Complexity and Interpretability	Individual decision trees are relatively easy to interpret as they represent decisions in a hierarchical structure.	While random forests improve performance, they can be more complex and less interpretable than single decision trees. However, they offer feature importance metrics and partial dependence plots to provide insights into the model's behavior.

- How does a random forest algorithm work?

Answer:

Random Forest is an ensemble learning algorithm that combines multiple decision trees to create a more robust and accurate model. It works as follows:-

- ***Random Selection of Features:*** When building each tree, a random subset of features is chosen at each split to reduce correlation between the trees.
    - ✓ *Bootstrap Sampling:* Generate multiple subsets of the training data using bootstrapping (sampling with replacement).
    - ✓ *Decision Trees:* For each subset, build a decision tree using a random subset of features at each split.
- ***Aggregation of Predictions:*** The final prediction is made by aggregating the predictions of all the individual trees, which helps in reducing variance and improving accuracy.
    - ✓ *Classification:* Use majority voting (the most common class label).
    - ✓ *Regression:* Use the average of the predictions from all trees.

- What is bootstrapping in the context of random forests?

Answer:

Bootstrapping is a statistical technique that involves sampling with replacement from the original dataset to create multiple subsets. In the context of random forests, it performs the following:-

- ***Sample Creation:*** Generate multiple bootstrap samples from the original training dataset. Each bootstrap sample is created by randomly selecting instances from the dataset with replacement, resulting in each sample having the same number of instances as the original dataset but with some instances potentially repeated.
- ***Model Training:*** Train individual decision trees on each bootstrap sample. Because each sample is different, the trees will vary, leading to diversity in the ensemble.

- ***Out – of – Bag Error:*** The instances not selected in a bootstrap sample (approximately 37% of the original dataset) can be used to estimate the out-of-bag (OOB) error, providing an unbiased estimate of the model's performance.

- Explain the concept of feature importance in random forests.

Answer:

Feature Importance in random forests refers to the contribution of each feature to the model's predictions. It is used to understand which features are most influential in making predictions.

It is calculated as follows:-

- ***Mean Decrease in Impurity (MDI):*** For each feature, measure how much the feature reduces the impurity (e.g., Gini impurity) in the trees where it is used. The importance score is the average of these reductions across all trees.
- ***Mean Decrease in Accuracy (MDA):*** Randomly permute the values of each feature and measure the decrease in the model's accuracy. A larger decrease indicates higher importance.

Usage of Feature Importance in Random Forests:-

- ***Model Importance:*** Identify the most important features that influence the predictions.
- ***Feature Selection:*** Select the most relevant features for building simpler and more efficient models.

- What are the key hyperparameters of a random forest and how do they affect the model?

Answer:

The key hyperparameters of a Random Forest are as follows:-

- ***Number of Trees:***
  - ✓ *Description:* The number of decision trees in the forest.
  - ✓ *Effect:* Increasing the number of trees generally improves performance but also increases computational cost.
  - ✓ *Representation:* In code, it is represented as 'n_estimators'.
- ***Maximum Depth:***
  - ✓ *Description:* The maximum depth of each tree.
  - ✓ *Effect:* Controls the complexity of the trees. Deeper trees can capture more information but may lead to overfitting.
  - ✓ *Representation:* In code, it is represented as 'max_depth'.
- ***Minimum Samples Split:***

- ✓ *Description:* The minimum number of samples required to split an internal node.
- ✓ *Effect:* Higher values prevent the model from learning overly specific patterns, reducing overfitting.
- ✓ *Representation:* In code, it is represented as 'min_samples_split'.

- ➢ ***Minimum Samples Leaf:***
  - ✓ *Description:* The minimum number of samples required to be at a leaf node.
  - ✓ *Effect:* Ensures that leaf nodes contain enough data, reducing overfitting.
  - ✓ *Representation:* In code, it is represented as 'min_samples_leaf'.
- ➢ ***Maximum Features:***
  - ✓ *Description:* The number of features to consider when looking for the best split.
  - ✓ *Effect:* Controls the randomness of the trees. Lower values increase diversity among trees, improving generalization.
  - ✓ *Representation:* In code, it is represented as 'max_features'.
- ➢ ***Bootstrap:***
  - ✓ *Description:* Whether to use bootstrapping when building trees.
  - ✓ *Effect:* If set to True, each tree is built on a bootstrap sample of the data. If False, the entire dataset is used.
  - ✓ *Representation:* In code, it is represented as 'bootstrap'.
- ➢ ***Criterion:***
  - ✓ *Description:* The function used to measure the quality of a split (e.g., Gini impurity or entropy for classification).
  - ✓ *Effect:* Affects how the trees are built and can influence the model's performance.
  - ✓ *Representation:* In code, it is represented as 'criterion'.
- ➢ ***Random State:***
  - ✓ *Description:* Seed for the random number generator.
  - ✓ *Effect:* Ensures reproducibility of results.
  - ✓ *Representation:* In code, it is represented as 'random_state'.

The impact of key hyperparameters on the model, can be summarized as follows:-

- ➢ ***Performance:*** Proper tuning of these hyperparameters can significantly improve model performance.
- ➢ ***Overfitting/Underfitting:*** Controls overfitting (too complex model) and underfitting (too simple model) by balancing tree depth, minimum samples, and the number of features considered.
- ➢ ***Computational Efficiency:*** Affects the training time and memory usage, especially the number of trees and their depth.

- Describe the logistic regression model and its assumptions.

Answer:

The Logistic Regression Model along with its assumptions has been described as follows:-

- ***Overview:***
  - ✓ Logistic regression is a statistical model used for binary classification problems.
  - ✓ It models the probability that a given input belongs to a particular class.
  - ✓ The model uses a logistic function (sigmoid function) to map the predicted values to probabilities.
- ***Model Equation:*** $P(Y = 1|X) = 1/(1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)})$

  where,

  $P(Y = 1|X)$ → Probability of the positive class

  X → Feature Vector

  B → Model Coefficients
- ***Assumptions:***
  - ✓ *Linearity:* The relationship between the features and the log-odds of the outcome is linear.
  - ✓ *Independence:* Observations are independent of each other.
  - ✓ *No Multicollinearity:* Features should not be highly correlated with each other.
  - ✓ *Sufficient Sample Size:* The dataset should have a sufficient number of observations.

- How does logistic regression handle binary classification problems?

Answer:

Logistic regression estimates the probability that an instance belongs to the positive class (class 1). A threshold (commonly 0.5) is used to classify instances:

- ✓ If $P(Y = 1|X) \geq 0.5$, classify as class 1.
- ✓ If $P(Y = 1|X) < 0.5$, classify as class 0.

  The logistic function ensures that the output probabilities are between 0 and 1.

- What is a sigmoid function and how is it used in logistic regression?

Answer:

The Sigmoid function is a mathematical function that produces an "S"-shaped curve, often used in machine learning, particularly in logistic regression and neural networks. It maps any real-valued number into a value between 0 and 1, making it useful for models that output probabilities or binary classification.

Mathematical Formula:

$$\sigma(x) = 1/(1+e^{-x})$$

where,

$\sigma(x) \rightarrow$ Sigmoid Function

$x \rightarrow$ Input to function (a real number)

$e \rightarrow$ Base of natural logarithm, approx. = 2.718

It can also be written as:

$$\sigma(z) = 1/(1+e^{-z})$$

where,

$\sigma(z) \rightarrow$ Sigmoid Function

$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$

Usage in Logistic Regression:

- The sigmoid function maps the linear combination of input features (the logit) to a probability between 0 and 1.
- It transforms the linear output into a nonlinear one, allowing for a probabilistic interpretation.

- Explain the concept of the cost function in logistic regression.

Answer:

The concept of the cost function in logistic regression has been explained as follows:-

- ***Overview:*** The cost function measures the error between the predicted probabilities and the actual class labels. In logistic regression, the cost function used is the log-loss or binary cross-entropy.
- ***Log – Loss Function:*** It can be defined as –

$$J(\beta) = -\frac{1}{m}\Sigma^{m}_{I=1}\, [y_i \log (h_\beta (x_i)) + (1 - y_i) \log (1 - h_\beta(x_i))]$$

  where,

  $h_\beta (x_i)$ → Predicted Probability for instance i

  y_i → Actual class label

  m → Number of training examples
- ***Purpose:*** The cost function is minimized using optimization algorithms like gradient descent to find the optimal model parameters β.

- How can logistic regression be extended to handle multiclass classification?

Answer:

Logistic regression can be extended to multiclass classification using either of the given strategy:-

- ***One-vs-Rest (OvR):***
  - ✓ Train separate binary classifiers for each class.
  - ✓ For each classifier, treat the current class as the positive class and all other classes as the negative class.
  - ✓ During prediction, select the class with the highest predicted probability.
- ***Softmax Regression (Multinomial Logistic Regression):***
  - ✓ Generalizes logistic regression to multiple classes using the softmax function.
  - ✓ The softmax function is defined as –

  $$P(Y = i|X) = (e^{\beta_i \cdot X})/(\Sigma^{k}_{j=1}\, e^{\beta_j \cdot X})$$

    where,

    k → Number of classes

    β_i → Coefficients for class i

    X → Feature Vector
  - ✓ The model is trained to minimize the cross-entropy loss for multiple classes.

- What is the difference between L1 and L2 regularization in logistic regression?

Answer:

The difference between L1 and L2 regularization in logistic regression are as follows:-

L1 Regularization	***L2 Regularization***
Adds the absolute value of the coefficients as a penalty term to the cost function.	Adds the squared value of the coefficients as a penalty term to the cost function.
Cost function with L1 regularization: $J(\beta) = -\frac{1}{m}\Sigma^{m}_{I=1}[y_i \log(h_\beta(x_i)) + (1 - y_i) \log(1 - h_\beta(x_i))] + \lambda\Sigma^{n}_{j=1} \lvert\beta_j\rvert$	Cost function with L2 regularization: $J(\beta) = -\frac{1}{m}\Sigma^{m}_{I=1}[y_i \log(h_\beta(x_i)) + (1 - y_i) \log(1 - h_\beta(x_i))] + \lambda\Sigma^{n}_{j=1} \beta^2{}_j$
Leads to sparse models with few features having non-zero coefficients.	Leads to small but non-zero coefficients for all features, reducing the impact of less important features without eliminating them entirely.

- What is XGBoost and how does it differ from other boosting algorithms?

Answer:

XGBoost (Extreme Gradient Boosting) is an optimized and scalable implementation of gradient boosting that is designed for speed and performance. It is widely used in machine learning competitions and real-world applications due to its high efficiency and predictive accuracy.

The key differences between XGBoost and other boosting algorithms are as follows:-

- ***Regularization:*** XGBoost includes L1 and L2 regularization, which helps in reducing overfitting and improving model generalization.
- ***Parallel Processing:*** XGBoost can perform parallel computations during the construction of trees, significantly speeding up the training process.
- ***Tree Pruning:*** XGBoost employs a technique called "max depth pruning" (also known as "depth-wise" pruning), which stops splitting once the maximum depth is reached, thus improving efficiency.
- ***Handling Missing Values:*** XGBoost has a built-in mechanism to handle missing values by learning the best direction to take when encountering missing data during training.
- ***Weighted Quantile Sketch:*** This allows handling of weighted data more effectively, providing better approximations for data with non-uniform weights.
- ***Scalability:*** XGBoost is designed to handle large-scale datasets efficiently and can be distributed across clusters.

- Explain the concept of boosting in the context of ensemble learning.

Answer:

Boosting is a popular ensemble learning technique designed to improve the accuracy and robustness of machine learning models. It works by combining the predictions of several weak learners to create a strong learner.

Here's a detailed explanation of the concept:-

- ***Weak Learners:***
  - ✓ *Definition:* Weak learners are models that perform slightly better than random guessing. They are typically simple models, such as shallow decision trees.
  - ✓ *Purpose:* Boosting focuses on improving the performance of these weak learners by combining them into a more powerful model.
- ***Sequential Learning:***
  - ✓ *Process:* Boosting builds models sequentially, with each new model correcting the errors made by the previous models.
  - ✓ *Error Correction:* Each subsequent model in the sequence focuses on the errors (misclassified examples) of the preceding models.
- ***Weighted Voting:***
  - ✓ *Model Contribution:* Each model's contribution to the final prediction is weighted based on its accuracy. Models that perform better have higher weights.
  - ✓ *Aggregation:* The final prediction is typically made by aggregating the predictions of all the models, with each model's prediction weighted according to its performance.
- ***Iterative Approach:***
  - ✓ *Training:* Boosting trains models iteratively, where each iteration improves upon the mistakes of the previous models.
  - ✓ *Update Weights:* In each iteration, the weights of the training examples are updated. Misclassified examples receive higher weights, making them more important for the next model.

Key steps in Boosting includes the following:-

- ***Initialization:*** Start with a base model, usually a simple model that provides initial predictions.
- ***Compute Residuals:*** Calculate the residuals (errors) of the current model's predictions. Residuals indicate the difference between the predicted values and the true values.
- ***Train New Model:*** Train a new model to predict the residuals from the previous step. This model focuses on the errors of the previous model.

- ***Update Model:*** Combine the new model with the existing ensemble. The predictions of the new model are weighted and added to the previous predictions.
- ***Repeat:*** Repeat the process for a specified number of iterations or until the model performance stops improving.
- ***Final Prediction:*** Aggregate the predictions of all models, using weighted voting (for classification) or weighted averaging (for regression) to make the final prediction.

Some of the types of Boosting Algorithms include the following, which have been represented in a tabular form, as follows:-

Boosting Algorithms	***Concept***	***Process***
AdaBoost (Adaptive Boosting)	Adjusts the weights of misclassified examples to focus more on hard-to-classify instances.	Each subsequent model corrects the mistakes of the previous model by adjusting the weights of misclassified instances.
Gradient Boosting	Fits new models to the residual errors of the existing ensemble.	Each model is trained to minimize the residual error of the previous models, effectively reducing the loss function.
XGBoost (Extreme Gradient Boosting)	An optimized implementation of gradient boosting with additional features such as regularization, parallel processing, and efficient handling of large datasets.	XGBoost initializes with an initial prediction, then iteratively fits decision trees to residuals. Gradients guide updates, and regularization prevents overfitting. The process repeats, combining trees to improve accuracy. It leverages parallel processing and handles missing data efficiently, resulting in a powerful, optimized gradient boosting model.

Advantages of Boosting Algorithm includes the following:-

- ***Improved Accuracy:*** By combining multiple weak learners, boosting can significantly improve model accuracy and robustness.
- ***Error Reduction:*** Boosting effectively reduces bias and variance, making it a powerful method for improving performance.
- ***Flexibility:*** Can be applied to various base models and adapted to different types of data and problems.

Disadvantages of Boosting Algorithm includes the following:-

- ***Computationally Intensive:*** Boosting can be computationally expensive and time-consuming due to the iterative training process.
- ***Overfitting:*** Although boosting reduces bias, it can sometimes lead to overfitting if not properly tuned, especially with complex models.
- ***Interpretability:*** The final model can be complex and less interpretable compared to simpler models.

- How does XGBoost handle missing values?

Answer:

XGBoost handles missing values by learning the optimal path (either left or right) to take when a missing value is encountered during training. This is achieved by:

- Assigning an instance with a missing value to either the left or right child node based on which choice results in a better split according to the loss function.
- During the training process, XGBoost finds the best way to handle missing values, thus making the model robust to missing data without requiring imputation.

- What are the key hyperparameters in XGBoost and how do they affect model performance?

Answer:

The key hyperparameters in XGBoost and their affect on model performance have been discussed in a tabular format, which is as follows:-

Key Hyperparameter	Description	Effect
Learning Rate (‘eta’)	Shrinks the contribution of each tree by a factor.	Lower values can improve model performance by making the model more robust, but require more trees to be built.
Number of Trees (‘n_estimators)	Number of boosting rounds or trees to be built.	More trees generally improve performance but also increase computation time and risk of overfitting.
Maximum Depth (‘max_depth)	Maximum depth of a tree.	Deeper trees can capture more information but may lead to overfitting. Shallower trees might underfit.

Subsample (‘subsample’)	Fraction of training instances to use for growing each tree.	Reducing the subsample can prevent overfitting but too small a value might lead to underfitting.
Column Subsample (‘colsample_bytree’ and ‘colsample_bylevel)	Fraction of features to consider for each tree and each level, respectively.	Helps in preventing overfitting and improves model diversity.
Regularization Parameters (‘lambda’ and ‘alpha’)	L2 (ridge) and L1 (lasso) regularization terms on weights.	Helps in preventing overfitting by penalizing large weights.
Minimum Child Weight (‘min_child_weight’)	Minimum sum of instance weight (hessian) needed in a child.	Higher values prevent overfitting by making the algorithm more conservative.
Gamma (‘gamma’)	Minimum loss reduction required to make a further partition on a leaf node.	Higher values make the algorithm more conservative, leading to fewer splits.

- Describe the process of gradient boosting in XGBoost.

Answer:

The process of Gradient Boosting in XGBoost can be described as follows:-

- ***Initialization:*** Start with an initial model, typically predicting the mean of the target variable.
- ***Compute Residuals:*** Calculate the residuals (errors) of the current model predictions.
- ***Fit a Tree:*** Train a new decision tree to predict the residuals (gradients) from the previous step.
- ***Update Model:*** Add the predictions from the new tree to the current model predictions, scaled by a learning rate.
- ***Repeat:*** Repeat the 2nd, 3rd and 4th steps for a specified number of iterations or until convergence.
- ***Mathematical Formulation:*** At each iteration t, the model $F_t(x)$ is updated as –

$$F_t(x) = F_{t-1}(x) + \eta . h_t(x)$$

where,

$\eta \rightarrow$ Learning Rate

$h_t(x) \rightarrow$ New tree trained to predict the residuals.

- What are the advantages and disadvantages of using XGBoost?

Answer:

Advantages:- The advantages of using XGBoost are as follows:

- ***High Performance:*** XGBoost often achieves better performance compared to other algorithms due to its robust handling of various data types and relationships.
- ***Speed:*** Efficient implementation with support for parallel and distributed computing.
- ***Handling Missing Values:*** Built-in support for handling missing data.
- ***Regularization:*** Built-in regularization terms prevent overfitting.
- ***Flexibility:*** Supports various objective functions and custom loss functions.
- ***Feature Importance:*** Provides insights into feature importance.

Disadvantages:- The disadvantages of using XGBoost are as follows:

- ***Complexity:*** More complex and harder to tune compared to simpler models like logistic regression or decision trees.
- ***Computationally Intensive:*** Can be resource-intensive, especially with large datasets.
- ***Overfitting:*** Although it includes mechanisms to prevent overfitting, it can still overfit if not properly tuned.
- ***Interpretability:*** Less interpretable compared to simpler models like decision trees or linear regression.

MACHINE LEARNING PRACTICAL QUESTIONS

Question No. – 1:

Take any project from PW Experience Portal form machine learning domain and make an end to end project with all the necessary documents.*

Solution:

Since, this question is a coding question and is an end-to-end Project, the code has been written in a modular fashion, consisting of multiple files, each meant for a particular function, performing a specified task and has been uploaded to GitHub.

The required Links are:

- ***Pre – Requisite:*** The basic links required, in order to begin this project are –
  - ✓ *PW Skills Experience Portal Website URL:* <https://experience.pwskills.com>
  - ✓ *Project Description Link:* <https://drive.google.com/file/d/1Vy9Su-RefQM__rLpFVqhJpCqrJzbdS_V/view>
  - ✓ *Dataset Link:* <Thyroid Disease - UCI Machine Learning Repository>
- ***Project Documentation:*** This consists of the GitHub as well as Google Drive Links of various documents related to this project. These include –
  - ✓ *High Level Document (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/HLD.pdf>
  - ✓ *High Level Document (Google Drive):* <https://drive.google.com/file/d/1xhbp7Z8uksAkdYZ5X1ugO_L4S1oDV1f6/view?usp=sharing>
  - ✓ *Low Level Document (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/LLD.pdf>
  - ✓ *Low Level Document (Google Drive):* <https://drive.google.com/file/d/1Pb-OTry_cjtXSi4U0VgewHQvsVw7Ond4/view?usp=sharing>
  - ✓ *Architecture Document (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/Architecture_Document.pdf>
  - ✓ *Architecture Document (Google Drive):* <https://drive.google.com/file/d/1RBVfpp9GlTgiuNw5Xr0vSC48LBCw4Pxm/view?usp=sharing>
  - ✓ *Wireframe Document (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-

Project/blob/main/Project_Documentation/Wireframe_Document.pdf
>

- ✓ *Wireframe Document (Google Drive):* <https://drive.google.com/file/d/1HPeTaye0lXRo6rAJ96KazYIzX0hII1N9/view?usp=sharing>
- ✓ *Detailed Project Report (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/Detailed_Project_Report.pdf>
- ✓ *Detailed Project Report (Google Drive):* <https://drive.google.com/file/d/1vel8TO3Ac6SRFjCiWNg4Ri7Rfyjao8nM/view?usp=sharing>
- ✓ *Project Demo Video (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/Project_Demo_Video.mp4>
- ✓ *Project Demo Video (Google Drive):* <https://drive.google.com/file/d/1in_P5o-T-u2h8H8c6GENVn_wYYlLajMy/view?usp=sharing>

➢ ***Project:*** This project has been completed and has been uploaded to GitHub as well as Google Drive. The required links are –

- ✓ *GitHub Repository Link:* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project.git>
- ✓ *Google Drive Link:* <https://drive.google.com/drive/folders/1grMSD8MWV5LQDfPiP2m5Pkbb36TaX8Ln?usp=drive_link>

➢ ***Letters / Certificates:*** Since, I have enrolled in the Project successfully and it has been completed successfully, both the Offer Letter and Internship Experience Letter/Certificate have been earned. The required links are –

- ✓ *Offer Letter (URL):* <https://internship-cdn.pwskills.com/letters/fb23f20d-063b-4712-ad07-b44269ec3379.pdf>
- ✓ *Offer Letter (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/Offer%20Letter.pdf>
- ✓ *Offer Letter (Google Drive):* <https://drive.google.com/file/d/1w_7CHRhpmvm7qLFqINTcImI5D9ySyv_1/view?usp=sharing>
- ✓ *Internship Letter/Certificate (URL):* <https://internship-cdn.pwskills.com/certificates/f0e33c01-c1e0-428e-a411-8d474db31a22.pdf>
- ✓ *Internship Letter/Certificate (GitHub):* <https://github.com/ImAni07/Thyroid-Disease-Detection-ML-Project/blob/main/Project_Documentation/Internship%20Experience%20Letter.pdf>

- ✓ *Internship Letter/Certificate (Google Drive):* <https://drive.google.com/file/d/1_xrgo4WltowIXcfr6mBLX5XyT7UhFwiY/view?usp=sharing>

➢ ***LinkedIn:*** The project has also been uploaded to LinkedIn, as required during submission of the project in PW Skills Experience Portal. The required links –

- ✓ *Post Link:* <https://www.linkedin.com/posts/anirban-majumder-49344a288_datascience-eda-machinelearning-activity-7231965382788403200-Uhcr?utm_source=share&utm_medium=member_desktop>
- ✓ *Profile Link:* <https://www.linkedin.com/in/anirban-majumder-49344a288/>

The basic Project Description is given as:-

Project Title	*Thyroid Disease Detection*
Technologies	*Machine Learning Technology*
Domain	*Healthcare*
Project Difficulties level	*Intermediate*

Question No. – 2:

Do the EDA on the given dataset: Lung cancer and extract some useful information from this. *

Dataset Description: Lung cancer is one of the most prevalent and deadly forms of cancer worldwide, presenting significant challenges in early detection and effective treatment. To aid in the global effort to understand and combat this disease, we are excited to introduce our comprehensive Lung Cancer Dataset.

Solution:

This is a coding question, with an objective of performing EDA on a given dataset of Lung Cancer. The codes have been written in a Jupyter Notebook and has been uploaded to GitHub.

The required links include the following:-

➢ ***Dataset Link:*** < lungs_cancer.zip - Google Drive>

➢ ***GitHub Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>

➢ ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Machine_Learning_Codes>

- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Machine_Learning_Codes/Lung_Cancer_EDA_%26_Prediction_with_ML.ipynb>

Question No. – 3:

Do the Eda on this Dataset :Presidential Election Polls 2024 Dataset and extract useful information from this.*

Dataset Description: This dataset comprises the results of a nationwide presidential election poll conducted on March 4, 2024. The data offers various insights but does not align with the official election results. You are encouraged to create your notebooks and delve into the data for further exploration.

Solution:

This is a coding question, with an objective of performing EDA on a given dataset of Lung Cancer. The codes have been written in a Jupyter Notebook and has been uploaded to GitHub.

The required links include the following:-

- ***Dataset Link:*** < election2024.csv - Google Drive>
- ***GitHub Repository Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment.git>
- ***Folder Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/tree/main/Solutions_AnswersAndCodes/Machine_Learning_Codes>
- ***Code Link:*** <https://github.com/ImAni07/Decode_Data_Science_with_Machine_Learning_Assignment/blob/main/Solutions_AnswersAndCodes/Machine_Learning_Codes/RussianElection2024_EDA_Code.ipynb>
