Project Title:

Books and Authors

Project Description:

This Python code defines three classes: Book for general books, Fiction for fiction books, and NonFiction for non-fiction books. Books are initialized with attributes like title, author, and pages. Fiction books additionally have a genre attribute, while non-fiction books have a topic attribute. The code demonstrates inheritance, encapsulation, abstraction, and polymorphism. It creates instances of these classes and showcases their attributes through respective methods.

Objectives:

- To examine survey information obtained from students about their favorite novels.
- To determine which authors, genres, and book lengths are popular with students.
- To produce reports and visualizations that offer practical advice for book promotion tactics.
- To design an intuitive user interface for uploading survey data and seeing the outcomes of analysis.

Importance and Contribution:

This task shows how object-oriented programming (OOP) works. It's all about important things like inheritance, storing data in groups, and customizing parts of the code. This helps with using code again and dealing with data. It also allows for specific changes and options, so different book types can work in the same way. In general, this code helps with using OOP ideas in real situations.

Hardware & Software Used:

Hardware:

- ✓ Computer

Software:

- ✓ Online GDB

Four Principles of Object-Oriented Programming

Inheritance:

```
class Fiction(Book):
class NonFiction(Book):
```

- Shows that "Fiction" and "NonFiction" inherit attributes and methods from the `Book` class. This means `Fiction` and `NonFiction` can access and use all functionalities of
  `Book` while also having their own unique features.

Polymorphism:

```
def display_info(self):
```

- Let's demonstrate polymorphism by allowing various classes to have methods with the same name but distinct implementations. This allows objects of those types to be managed uniformly with a single interface.

Encapsulation:

```
def __init__(self, title, author, pages):
def __init__(self, title, author, pages, genre):
```

- Enhanced encapsulation through class-specific initialization and bundling of object characteristics. This promotes data integrity and abstraction by guaranteeing that the internal state of the objects is hidden and accessible only through specific ways.

Abstraction:

```
def display_info(self):
```

- Represents abstraction by concealing the technical implementation details of the information presentation and offering a simpler interface for retrieving book information. It enables interaction between items and people without requiring them to comprehend the underlying complexity.

Code Documentation:

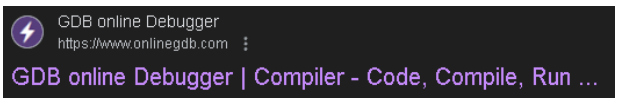
```
class Book:
    def __init__(self, title, author, pages):  # Encapsulation: Constructor to initialize object
        state self._title = title
        self._author = author
        self._pages = pages

 def display_info(self):  # Abstraction: Exposes only necessary features to interact with
the class
        print(f"Title: {self._title}")
        print(f"Author:
        {self._author}")
        print(f"Pages:
        {self._pages}")

class Fiction(Book):  # Inheritance: Fiction class inherits from Book
    class def __init__(self, title, author, pages, genre):
        super().__init__(title, author, pages)
        self._genre = genre

    def display_genre(self):  # Polymorphism: Method specific to Fiction class
        print(f"Genre: {self._genre}")

    # Polymorphism: Overriding the display_info
    method def display_info(self):
        super().display_info()
        self.display_genre()

class NonFiction(Book):  # Inheritance: NonFiction class inherits from Book
    class def __init__(self, title, author, pages, topic):
        super().__init__(title, author, pages)
        self._topic = topic

    def display_topic(self):  # Polymorphism: Method specific to NonFiction
        class print(f"Topic: {self._topic}")
```

```
  # Polymorphism: Overriding the display_info
    method def display_info(self):
        super().display_info()
        self.display_topic()

# Create objects using tuples
fiction_books = [
    ("And Then There Were None", "Agatha Christie", 200, "Mystery"),
    ("Harry Potter and the Philosopher's Stone", "J.K. Rowling", 320, "Fantasy")
]
nonfiction_info = ("Sapiens", "Yuval Noah Harari", 443, "History")

# Create objects and call
methods for info in fiction_books:
    fiction_book =
    Fiction(*info)
    fiction_book.display_info(
    ) print("-" * 20)

nonfiction_book = NonFiction(*nonfiction_info)
nonfiction_book.display_info()
```

User Guide:

Step 1:

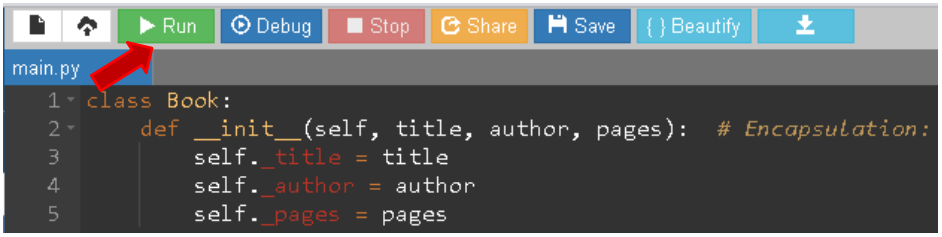
- First you need a website that can run the code:



https://www.onlinegdb.com/

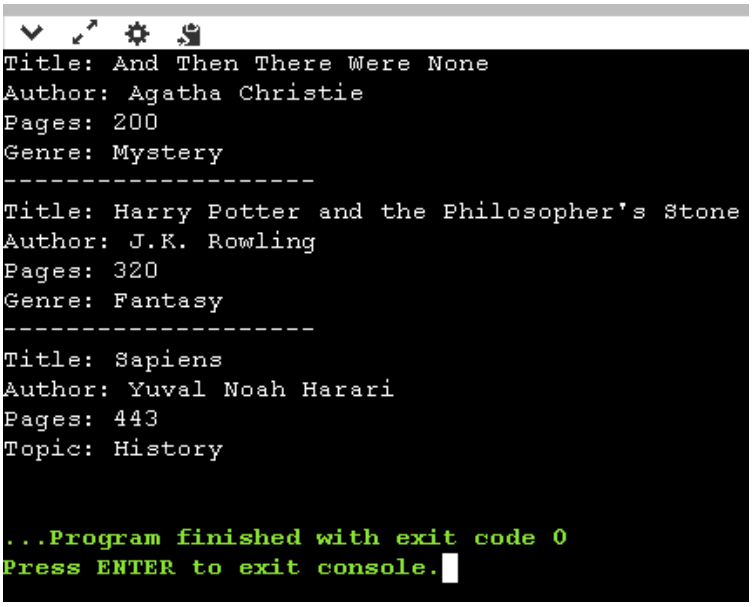
Step 2:

- Paste the code, then start the program click the "Run" button.



Step 3:

- At the bottom you'll see the program executed.



Description:

The classes for books (Book), fiction books (Fiction), and non-fiction books (NonFiction) are defined in the code. It displays abstract, polymorphism, inheritance, and encapsulation as OOP ideas. To show how book information is organized and managed, objects representing various book genres are made, and their details are printed.

Conclusion:

In general, this code successfully illustrates important concepts of Object-Oriented Programming (OOP), including abstraction, polymorphism, inheritance, and encapsulation. It demonstrates how to handle and organize data in a systematic and modular manner by creating classes for books, fiction books, and non-fiction books. The code demonstrates how object-oriented programming (OOP) may offer a lucid and efficient way to represent real-world things and their actions by means of object creation and class method use. All things considered, this example is a great teaching tool for learning and using OOP ideas in Python programming.

References:

Encapsulation: https://docs.python.org/3/tutorial/classes.html

Inheritance: https://docs.python.org/3/tutorial/classes.html

Polymorphism: https://stackoverflow.com/questions/12031018/overriding-in-python

super(): https://realpython.com/python-super/#an-overview-of-pythons-super-function

GDB code: https://www.onlinegdb.com/edit/HTSHEG8gc