Python

Presented by: Pranathi

Overview

- Functions
- Modules
- Data Manipulation

Introduction

Functions:

- A function is a block of code that performs a specific task.

Benefits:

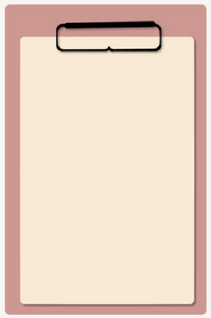
- Increase Code Readability
- Increase Code Reusability

Explanation

SUPPOSE, YOU NEED TO CREATE A PROGRAM TO CREATE A CIRCLE AND COLOR IT. YOU CAN CREATE TWO FUNCTIONS TO SOLVE THIS PROBLEM:

- **CREATE A CIRCLE FUNCTION**
- **CREATE A COLOR FUNCTION**

DIVIDING A COMPLEX PROBLEM INTO SMALLER CHUNKS MAKES OUR PROGRAM EASY TO UNDERSTAND AND REUSE.

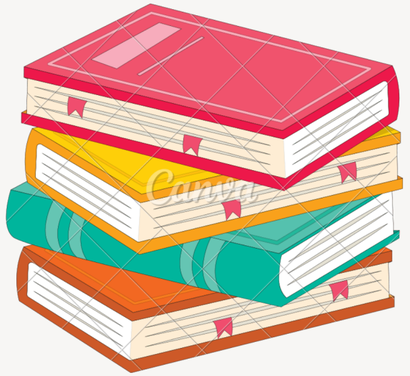


Types of Functions

- There are two types of function in Python programming:

Standard library functions - These are built-in functions in Python that are available to use.

User-defined functions - We can create our own functions based on our requirements.



Function Declaration

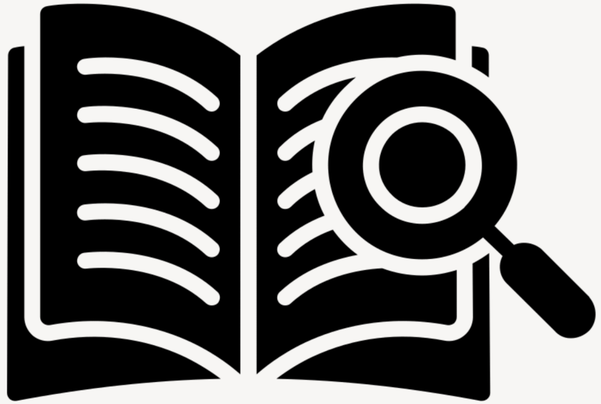
The syntax to declare a function is:

```
def function_name(arguments):
    # function body
    return
```

Here:

- def - keyword used to declare a function
- function_name - any name given to the function
- arguments - any value passed to function
- return (optional) - returns value from a function

Example

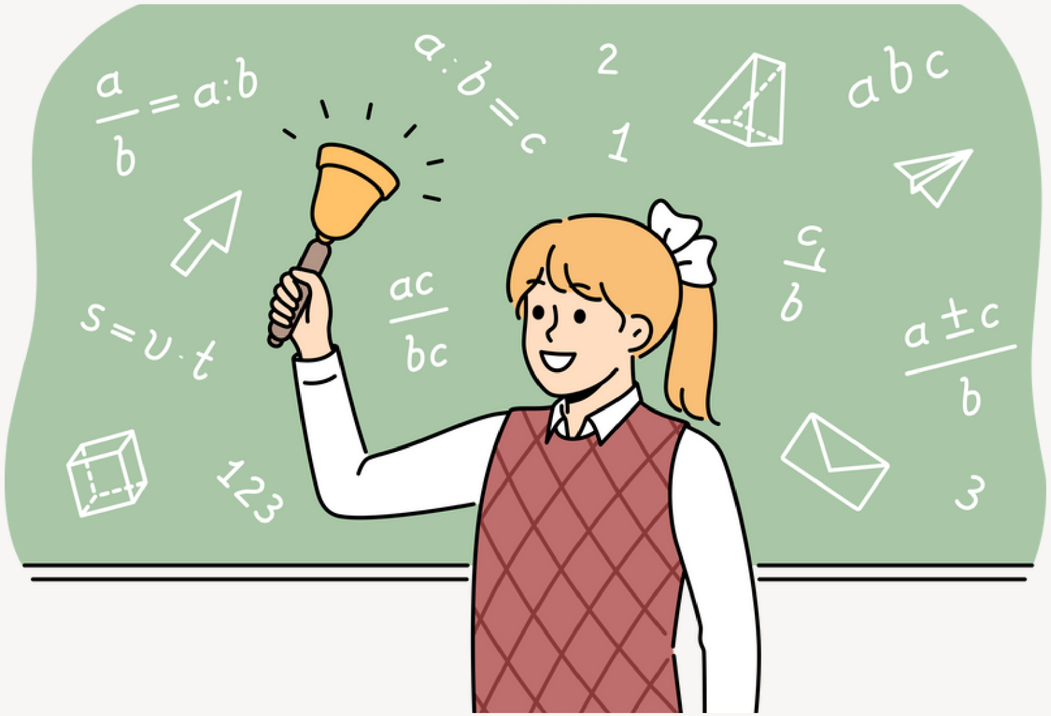


Python Function:

```
def greet():
    print('Hello World!')
# call the function
    greet()
print('Outside function')
```

Output:

```
Hello World!
Outside function
```



Introduction

Modules:

- Module is a file that contains code to perform a specific task. A module may contain variables, functions, classes etc

Benefits:

- Simpllicity
- Reusability

Explanation

As our program grows bigger, it may contain many lines of code. Instead of putting everything in a single file, we can use modules to separate codes in separate files as per their functionality. This makes our code organized and easier to maintain.

Import modules in Python

We can import the definitions inside a module to another module or the interactive interpreter in Python.We use the import keyword to do this. To import our previously defined module example, we type the following in the Python prompt.

Syntax:

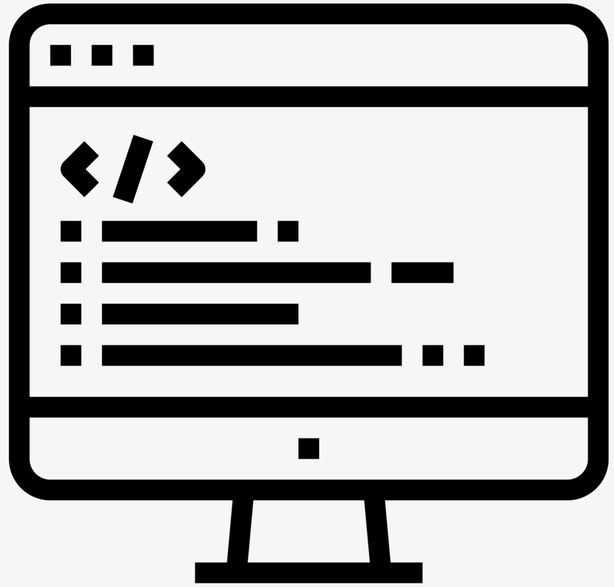
import example

Python from import statement:

We can import specific names from a module without importing the module as a whole.

For example:

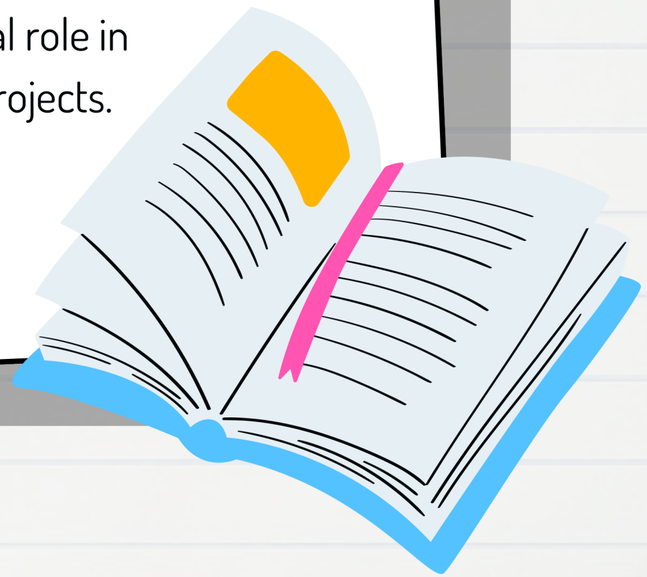
```
# import only pi from math module
from math import pi
print(pi)
# Output: 3.141592653589793
```



Introduction

Data Manipulation:

- Data manipulation is a fundamental concept in the field of data science and analysis. It refers to the process of changing, organizing, or transforming data to extract meaningful insights, make data more suitable for analysis, or prepare it for various applications. Data manipulation plays a crucial role in data preprocessing, which is the initial step in most data analysis projects.



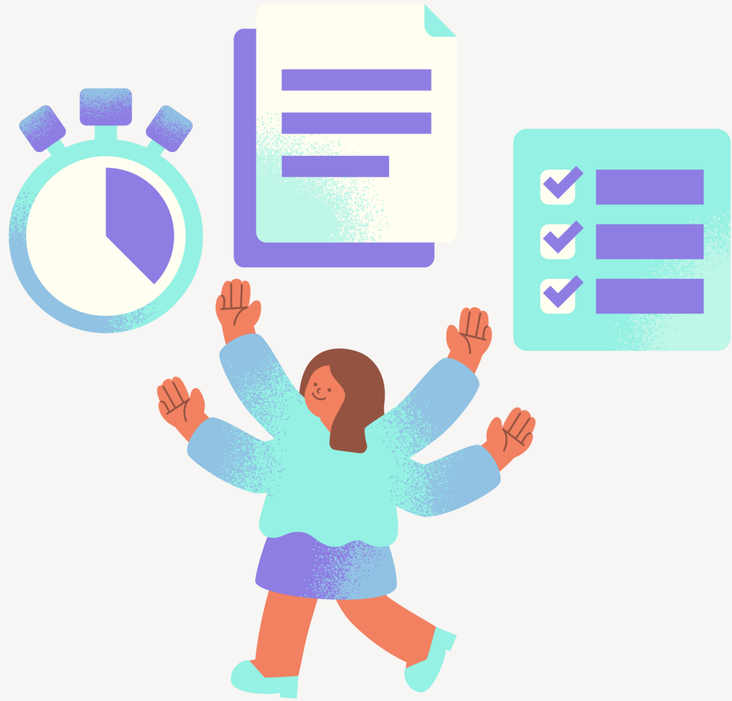
What is Data Manipulation?

- Data manipulation as the process of transforming, cleaning, and organizing data to make it suitable for analysis.

- It is a crucial step in data analysis.

Common Data Manipulation Tasks:

- Data Cleaning
- Data Transformation
- Data Agregation
- Merging and Joining Data
- Data Visualization
- Automation



Data Manipulation Libraries in Python

- Pandas
- NumPy
- Dask
- scikit-learn (for feature engineering)
- Matplotlib/Seaborn (for visualization)

Introduction to Pandas

- Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the Numpy library of Python.

Installing Pandas

```
pip install pandas
```

Import Pandas:

```
import pandas
```

Pandas for Data Manipulation

- **Data Exploration:**

  View first rows: “df.head()”
  Data summary: “df.info()”
  Basic statistics: “df.describe()”

- **Data Cleaning:**

  Handling missing values: “df.dropna(), df.fillna()”
  Removing duplicates: “df.drop_duplicates()”
  Correcting inaccurate data:” df.replace()”
  Standardizing data formats: “df.apply()”

- **Data Transformation**

  Create new columns: “**df['new_column'] = ...**”
  Group and aggregate data: “**df.groupby('column').agg()**”
  Pivot data:” **df.pivot_table()**”

- **Data Visualization**

  Basic plotting: “df.plot()”

- Functions allow us to encapsulate logic, promoting code reusability and maintainability.Modules provide a way to organize and reuse code across projects, fostering a more modular and efficient development process

- Python, equipped with powerful libraries like Pandas and NumPy, offers a robust environment for data manipulation and analysis.

- As you continue your Python journey, remember that mastering functions, modules, and data manipulation is an ongoing process.

Advantages

- Data Quality Management
- Data Integrity
- Efficient data storage
- Customization
- Automation
- Insight Discovery

Thank's For Watching