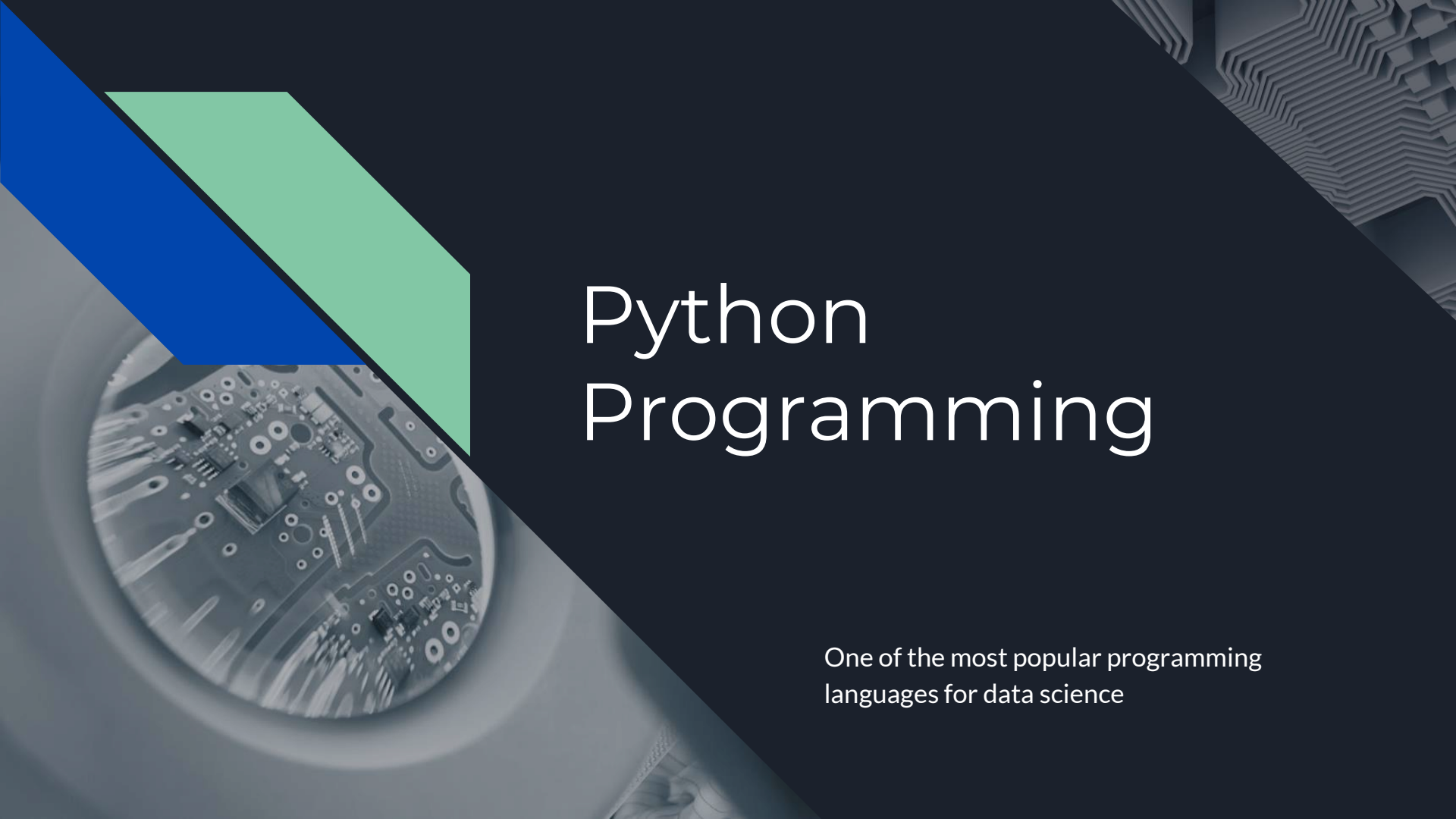


Python Programming

One of the most popular programming languages for data science

Introduction to Python

What is Python?

Difference between programming and scripting language.

History of Python.

Scope of Python.

Installing Python

Sample Code and Execution

What is Python?

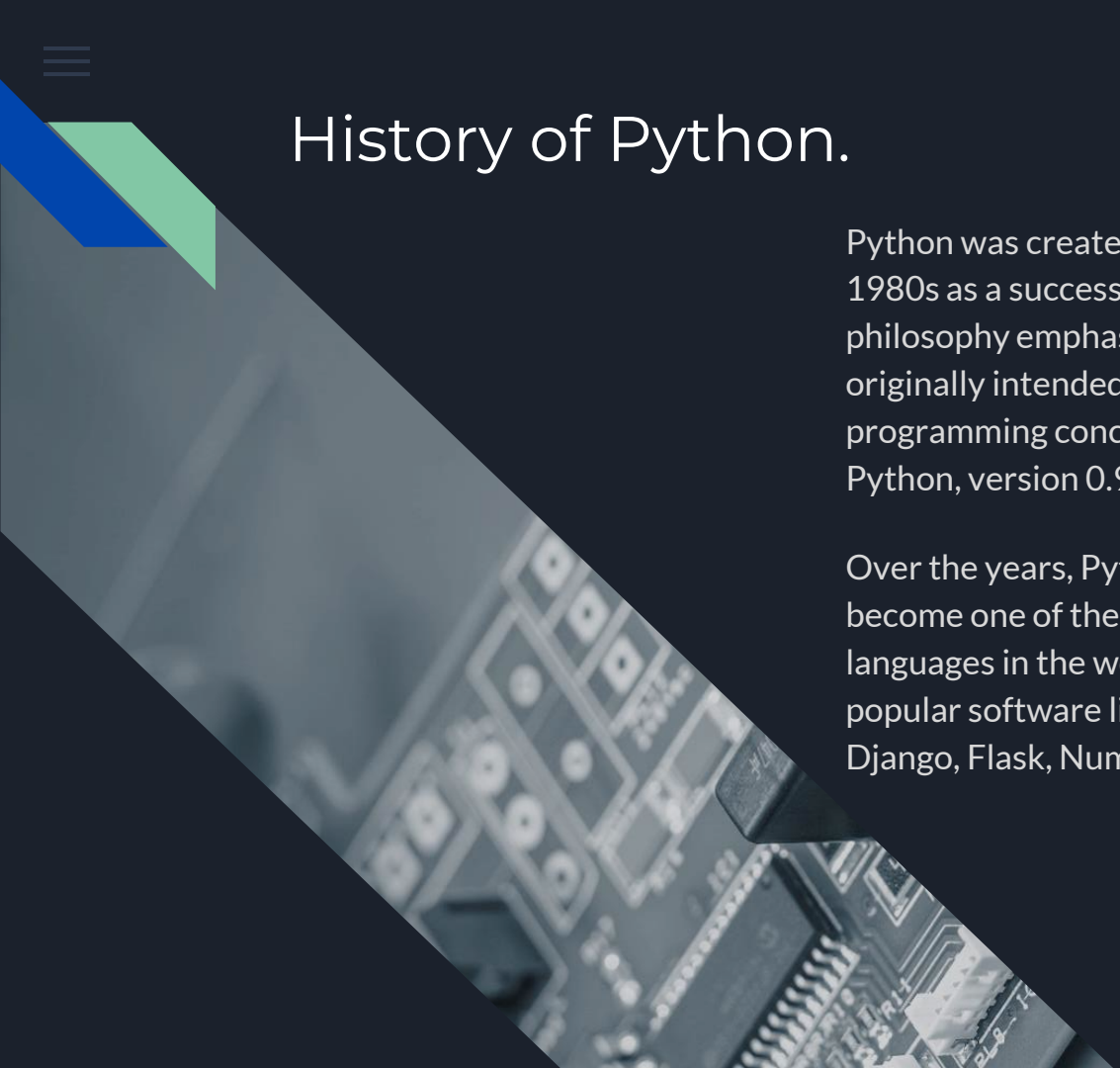
Python is a high-level, interpreted programming language that is widely used for a variety of purposes such as web development, scientific computing, data analysis, artificial intelligence, and more. It was created by Guido van Rossum and first released in 1991. Python is known for its simple and easy-to-read syntax, which makes it a popular choice for beginners and experienced programmers alike.

Difference between programming and scripting language.

Programming languages and scripting languages are both used to create software, but they differ in their approach and purpose. Programming languages, such as Java, C++, are used to write code that is compiled into executable programs. They tend to be more complex and powerful than scripting languages, and are often used for system-level programming, high-performance computing, and other applications that require low-level control.

Scripting languages, on the other hand, are interpreted on-the-fly and executed line-by-line. They tend to be simpler and more flexible than programming languages, and are often used for automation, web development, and other applications that require rapid prototyping and iteration.

History of Python.



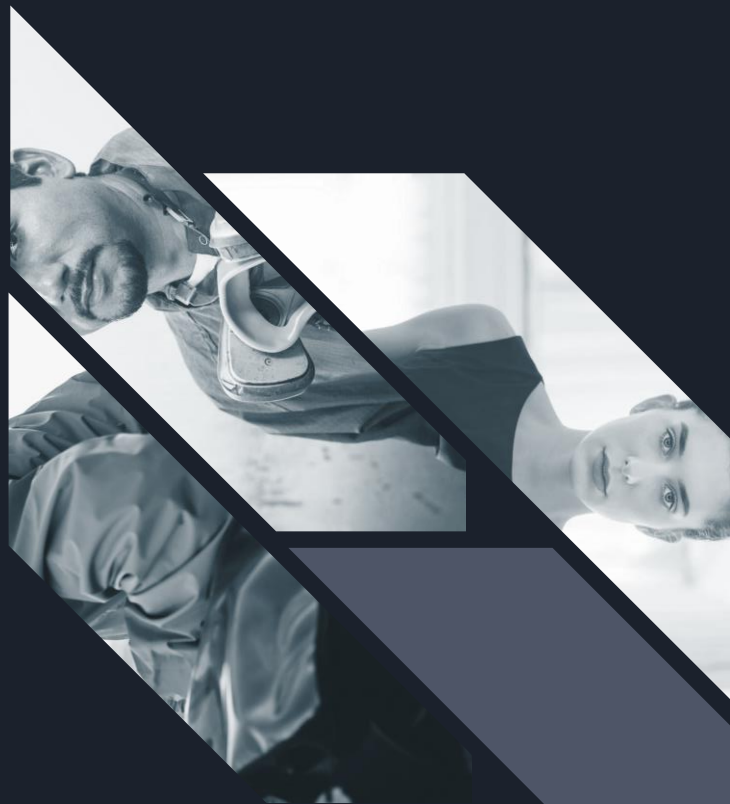
Python was created by Guido van Rossum in the late 1980s as a successor to the ABC language. Its design philosophy emphasized code readability, and it was originally intended to be a language for teaching programming concepts to beginners. The first version of Python, version 0.9.0, was released in 1991.

Over the years, Python has grown in popularity and become one of the most widely used programming languages in the world. It has been used to create many popular software libraries and frameworks, including Django, Flask, NumPy, and more.

Scope of Python.

Python has a wide range of applications and is used in various industries. Some of the areas where Python is commonly used include:

- Web development
- Scientific computing
- Data analysis
- Machine learning and artificial intelligence
- Game development
- System administration
- Network programming
- Desktop application development



Installing Python

To install Python, you can visit the official website at www.python.org and download the latest version of Python for your operating system. Python is available for Windows, Mac OS X, and Linux. Once you have downloaded the installation file, run it and follow the instructions to install Python on your computer

Assignments Day 1:

1. Open Python Interpreter and use it as calculator.
2. Explore operators like, +, -, *, /, %, **, ^ Create one text file and document use of each
3. Create hello.py file and print “Hello World”.
4. https://dev.to/arindamdawn/30-days-of-python-day-1-5ghh Go through day 1

How to Submit Assignments:

Create one repo named python_training in your github account. For assignments of every day create folder day1 and it should contains all solution files for that day.

For today you will have 2 files in folder

1. Operators.txt
2. hello.py

Python Variables and operators

- What are variables?
- Naming a Variable
- The Assignment Sign
- Basic Operators
- More Assignment Operators

What are variables?

Variables are containers that store values in Python. They are used to represent data such as numbers, strings, and other objects, and can be manipulated and updated as the program runs. To create a variable, you need to specify a name for it and assign a value to it.

Naming a Variable

When naming a variable in Python, there are a few rules to follow:

- Variable names can contain letters, numbers, and underscores, but cannot begin with a number.
- Variable names are case-sensitive, meaning "myVar" and "myvar" are two different variables.

Variable names should be descriptive and meaningful, and follow a consistent naming convention such as camelCase or snake_case.

Here are some examples of valid variable names in Python:

- age = 30
- name = "John"
- income_tax_rate = 0.25

The Assignment Sign

To assign a value to a variable in Python, use the equals sign (=). Here's an example:

```
x = 10
```

This creates a variable named "x" and assigns it the value of 10.

Basic Operators

In Python, you can perform basic mathematical operations on variables using arithmetic operators. Here are some examples:

```
x = 10

y = 5

z = x + y # Addition

w = x - y # Subtraction

q = x * y # Multiplication

r = x / y # Division
```

More Assignment Operators

In addition to the basic assignment operator (=), Python also supports several compound assignment operators that combine an arithmetic operator with the assignment operator. Here are some examples:

```
x = 10

x += 5 # Equivalent to x = x + 5

x -= 5 # Equivalent to x = x - 5

x *= 2 # Equivalent to x = x * 2

x /= 2 # Equivalent to x = x / 2
```

Day - 2 Assignments:

1. Create a variable called "age" and assign it the value 25.
2. Create a variable called "name" and assign it your name as a string.
3. Create a variable called "temperature" and assign it a float value representing the current temperature in degrees Celsius.
4. Calculate the area of a rectangle with length 5 and width 10, and assign the result to a variable called "area".
5. Increment the value of the "age" variable by 1 using a compound assignment operator.
6. Create function in python which take integer as input and convert it to binary string

   Input 5 → 101

   7 → 111

Python Data Types and Data Structures

- Integers
- Float
- String
- Type Casting In Python
- List
- Tuple
- Dictionary

Integers and Float

Integers are whole numbers, without any decimal points. They can be positive or negative. Here are some examples of integers in Python:

```
x = 10
y = -5
z = 0
```

Floats are numbers with decimal points. They can also be positive or negative. Here are some examples of floats in Python:

```
x = 3.14
y = -2.5
z = 0.0
```

String and Type Casting in Python

Strings are sequences of characters, enclosed in single or double quotes. They can contain letters, numbers, and symbols. Here are some examples of strings in Python:

```
x = "Hello, world!"
y = 'Python is fun'
z = "12345"
```

Type casting is the process of converting a value from one data type to another. In Python, you can use functions like int(), float(), and str() to convert between data types. Here are some examples:

```
x = 5 # integer
y = float(x) # convert integer to float
z = str(x) # convert integer to string
```

List and Tuple

A list is a collection of values, enclosed in square brackets and separated by commas. Lists can contain elements of different data types, and they can be modified (added, removed, or updated) during the program's execution. Here's an example of a list:

fruits = ["apple", "banana", "cherry"]

A tuple is similar to a list, but it is enclosed in parentheses instead of square brackets. Tuples are immutable, which means that their elements cannot be changed once they are created. Here's an example of a tuple:

numbers = (1, 2, 3, 4, 5)

Dictionary

A dictionary is a collection of key-value pairs, enclosed in curly braces and separated by commas. Dictionaries are used to store and retrieve data based on a unique key. Here's an example of a dictionary:

person = {"name": "John", "age": 30, "city": "New York"}

Day-3 Assignments

1. Create an integer variable and a float variable, and add them together. Check output type
2. Create a string variable containing your name, and concatenate it with a string variable containing your favorite color.
3. Convert a string variable containing the value "10" to an integer.
4. Create a list containing your favorite books.
5. Create a tuple containing the numbers 1, 3, and 5.
6. Create a dictionary containing the key-value pairs for your name, age, and hometown.

Day 3 Handson

List Manipulation:

Write a Python program that takes a list of integers as input and performs the following operations:

- Remove the duplicates from the list.
- Sort the list in descending order.
- Calculate the sum of all the elements in the list.

Tuple Operations:

Create a tuple containing the names of five countries. Write a Python program that performs the following operations:

- Print the length of the tuple.
- Check if a given country is present in the tuple.
- Create a new tuple by concatenating the original tuple with another tuple containing five more countries.
- Extra: Try modifying the element at 2nd index. What is output and why. Discuss it.

Dictionary Manipulation:

Write a Python program that operates on a dictionary representing the stock of items in a store. The dictionary should contain items as keys and their corresponding quantities as values. Perform the following operations:

- Add a new item to the stock.
- Update the quantity of an existing item.
- Remove an item from the stock.
- Print the items in stock alphabetically sorted along with their quantities.

List Comprehensions:

Write a Python program that generates a list of squares of even numbers between 1 and 20 using list comprehension.

Dictionary Iteration:

Create a dictionary representing the population of five different cities. Write a Python program that prints the city with the highest population along with its population.

Tuple Unpacking:

Write a Python program that takes a tuple of three integers representing the sides of a triangle as input and determines if it forms a valid triangle. If it does, print its type (equilateral, isosceles, or scalene).

List Sorting:

Write a Python program that takes a list of tuples, where each tuple contains a student's name and their score in a test. Sort the list based on the scores in descending order and print the names of the top three students.

Dictionary Filtering:

Create a dictionary representing the prices of different items in a store. Write a Python program that filters out the items with prices less than a given threshold and prints them.

List Operations:

Write a Python program that takes two lists of integers as input and performs the following operations:

- Find the intersection of the two lists (common elements).
- Find the union of the two lists (all elements without duplicates).
- Find the elements present in the first list but not in the second list.

Dictionary Sorting:

Write a Python program that takes a dictionary containing names as keys and their corresponding ages as values. Sort the dictionary based on ages in ascending order and print the names of the oldest and youngest person.

Dict = {‘parth’: 23, ‘temp’: 24, ‘nikul’: 18}

4. Make Your Program Interactive

Input() and print() functions

- The input() function in Python is used to take input from the user. The input() function takes a string argument, which is displayed as a prompt to the user. Here's an example:
  - name = input("What is your name? ")
- The print() function in Python is used to display output to the user. You can pass one or more values to the print() function, and they will be displayed on the console. Here's an example: print(“Hello world”)

Triple Quotes and Escape Characters

- Triple quotes are used to enclose multi-line strings in Python. You can use single or double quotes for the triple quotes. Here's an example:
  - text = '''This is a multi-line string.
  - It contains multiple lines of text. '''
  - print(text)
- Escape characters are special characters that are used to format the output. They are preceded by a backslash () character. Here are some commonly used escape characters in Python:
  - \n : newline
  - \t : tab
  - \r : carriage return
  - ': single quote
  - ": double quote

Day 4 Assignments

1. Write a program that takes two numbers as input from the user and displays their sum.
2. Write a program that takes a string as input from the user and displays it in uppercase letters.
3. Write a program that displays the following text, using triple quotes:
   Programming is fun.
   I love Python.
1. Write a program that displays the following text, using escape characters:
   She said, "Hello, world!"
1. Write a program that takes a sentence as input from the user and displays the number of words in the sentence.

Day 4 handson

- Scenario: Bookstore Inventory Management
  - Problem: You are managing the inventory of a bookstore. You need to create a Python program that allows you to add new books to the inventory and display the current inventory.
  - Description:
    - Program should contain add_book function which ask user name of book and in second line quantity of book. Add it in inventory dictionary
    - display_inventory: Print all books currently in dict
    - update_book_quantity: input book_name and updated quantity and update in dictionary.
    - Application should keep running until user wants to exit
    - Main Options:
      - 1. Add a book to inventory
      - 2. Display current inventory
      - 3. Update book quantity
      - 4. Exit

5. Making choice and Decision

If Statement

The if statement in Python is used to execute a block of code only if a certain condition is true. The syntax of the if statement is as follows:

```
if condition:

  # block of code to execute if the condition is true
```

Inline If

- The inline if statement is a shorthand version of the if statement. It is used to execute a single line of code if a condition is true. The syntax of the inline if statement is as follows

```
value = 10

result = "even" if value % 2 == 0 else "odd"

print(result)
```

For and while Loop

The for loop in Python is used to iterate over a sequence of values. The syntax of the for loop is as follows:

for variable in sequence:

    # block of code to execute for each value in the sequence

While Loop

The while loop in Python is used to execute a block of code repeatedly as long as a certain condition is true. The syntax of the while loop is as follows:

while condition:

    # block of code to execute as long as the condition is true

Break and Continue

The break keyword in Python is used to exit a loop prematurely. When the break keyword is encountered in a loop, the loop is immediately terminated, and control is passed to the next statement after the loop.

Continue

The continue keyword in Python is used to skip the current iteration of a loop and move on to the next iteration. When the continue keyword is encountered in a loop, the current iteration is skipped, and the loop continues with the next iteration.

Try, Except

The try-except block in Python is used to handle exceptions. When an exception occurs in a block of code inside the try block, control is passed to the except block, which contains the code to handle the exception. The syntax of the try-except block is as follows:

```
try:
    # block of code that may raise an exception
except ExceptionType:
    # block of code to handle the exception
```

Assignments:

1. Write a program that takes a number as input from the user and displays whether the number is even or odd.
2. Write a program that takes a list of numbers as input from the user and displays the sum of all the even numbers in the list.
3. Write a program that displays the numbers from 1 to 10 using a for loop. (try with single line)
4. Write a program that takes a number as input from the user and displays the multiplication table of the number using a while loop.
5. Write a program that takes a list of numbers as input from the user and displays only the odd numbers in the list using a for loop.
6. Write a program that takes a number as input from the user and displays whether the number is prime or not using a try-except block.

Explore: https://www.geeksforgeeks.org/python-except ion-handling/
https://docs.python.org/3/tutorial/errors.html#

6. Function and Modules

What are Functions?

A function is a block of code that performs a specific task. Functions are useful for breaking down large programs into smaller, more manageable pieces. Functions can take inputs and return outputs, and they can also be called from within other functions.

Defining Your Own Functions

To define a function in Python, we use the def keyword followed by the name of the function and the input parameters in parentheses. The body of the function is indented and contains the instructions that are executed when the function is called.

```
def my_function(parameter1, parameter2):

  # instructions to be executed

  return result
```

Variable Scope

Variable scope refers to the part of the program where a variable can be accessed. In Python, variables can have global scope or local scope. Global variables can be accessed from anywhere in the program, while local variables can only be accessed within the function where they are defined.

https://www.geeksforgeeks.org/python-scope-of-variables/

Importing Modules

Python has a large number of built-in modules that provide useful functions and data structures. To use a module in a program, we need to import it using the import keyword. We can then use the functions and data structures defined in the module in our program.

```
import math

result = math.sqrt(25)
```

https://www.geeksforgeeks.org/import-module-python/
https://docs.python.org/3/tutorial/modules.html

Creating our Own Module

We can also create our own modules in Python. A module is simply a file containing Python code. To create a module, we simply write the code that we want to include in the module in a file with a .py extension.

Assignments:

1. Write a program that defines a function to calculate the area of a rectangle. The function should take the length and width of the rectangle as input parameters and return the area.
2. Write a program that defines a function to calculate the factorial of a number. The function should take a single input parameter and return the factorial of that number.
3. Write a program that defines a function to calculate the sum of a list of numbers. The function should take a list of numbers as input and return the sum of those numbers.
4. Write a program that defines a function to calculate the nth Fibonacci number. The function should take a single input parameter and return the nth Fibonacci number.
5. Create a module called my_module that defines a function to calculate the square of a number. Import the module into a new program and use the function to calculate the square of a number.

7. Working With Files

Opening and Reading Text Files

We can open a text file for reading using the open() function. The open() function takes two arguments: the name of the file and the mode in which the file should be opened. To read from a file, we should use the mode 'r'.

```
file = open('example.txt', 'r')
```

https://docs.python.org/3/library/functions.html#open
https://www.geeksforgeeks.org/file-handling-python/

Using a For Loop to Read Text Files

We can use a for loop to read a text file line by line. The readline() method reads a single line from the file and moves the file pointer to the next line.

```
file = open('example.txt', 'r')

for line in file:

    print(line)
```

Writing to a Text File

We can open a text file for writing using the open() function. To write to a file, we should use the mode 'w'. The write() method writes a string to the file.

```
file = open('example.txt', 'w')

file.write('This is a sample text.')

file.close()
```

Opening and Reading Text Files by Buffer Size

We can also read a text file by buffer size using the read() method. The read() method reads a specified number of characters from the file.

```
file = open('example.txt', 'r')

data = file.read(10)

print(data)
```

Deleting and Renaming Files

We can delete a file using the os.remove() function, and we can rename a file using the os.rename() function.

```
import os

os.remove('example.txt')

os.rename('example.bin', 'new_example.bin')
```

Assignments

1. Write a program that opens a text file and reads the contents of the file.
2. Write a program that opens a text file and writes some text to the file.
3. Write a program that opens a binary file and reads the first 10 bytes of the file.
4. Write a program that creates a new text file, writes some text to the file, and then renames the file.
5. Write a program that deletes a text file.

Final Assignments

Assignment 1: Word Counter

- Create a Python script that reads a text file and counts the frequency of each word in the file.
- Store the word frequencies in a dictionary.
- Print out the top 5 most common words and their frequencies.

Assignment 2: To-Do List Manager

- Create a Python script that implements a simple to-do list manager.
- Allow users to add tasks, mark tasks as completed, and remove tasks.
- Store the tasks in a list of dictionaries where each dictionary represents a task with keys like 'task_name', 'priority', 'completed', etc.
- Provide a menu-driven interface for users to interact with the to-do list.

Assignment 3: Contact Book

- Develop a Python program to manage a contact book.
- Allow users to add new contacts, search for contacts by name, and delete contacts.
- Use a dictionary to store contacts where the keys are contact names and the values are dictionaries containing contact information like phone number, email, etc.
- Implement error handling for cases where a contact is not found during search or deletion.

Assignment 4: File Encryption

- Create a Python script that encrypts and decrypts text files using a simple substitution cipher.
- Implement functions for encryption and decryption using a basic substitution cipher algorithm (e.g., shifting each letter by a fixed number of positions in the alphabet).
- Prompt users to enter a filename and choose whether to encrypt or decrypt the file.
- Ensure the script handles cases where the file does not exist or cannot be opened for reading/writing.

Assignment 5: Inventory Management System

- Design a Python program to manage an inventory system for a small store.
- The inventory should consist of items with attributes like name, quantity, price, and category.
- Implement functionality to add new items, update existing items, remove items, and display the inventory.
- Allow users to search for items by name or category and display details of the matching items.
- Include features for checking the availability of items, adding items to a shopping cart, and generating a bill.
- Ensure the program handles errors gracefully, such as attempting to remove a non-existent item or adding a negative quantity.
- Implement a simple text-based user interface with a menu system for navigating different functionalities.