**ASSIGNMENT9**

**TOPIC NAME:RANDOM MODULE**

**Introduction to Python Modules and Importing Modules**

In Python, a **module** is a file containing Python definitions and statements. A module allows you to organize your Python code in a modular way, making it reusable, easy to maintain, and better structured.

Modules can include functions, variables, and even classes that can be used in other programs. Python provides both **built-in modules** (like math, random, etc.) and **external modules** that you can install and use in your projects.

**1. Creating a Module**

A module is simply a Python file with a .py extension. For example, let's create a module called mymodule.py.

# mymodule.py

def greet(name):

print(f"Hello, {name}!")

def add(a, b):

return a + b

**2. Importing a Module**

To use the functions or variables from another Python file (module), you need to **import** that module into your script.

**Importing the entire module**

import mymodule

mymodule.greet("Alice")

result = mymodule.add(5, 3)

print(f"The sum is: {result}")

**Output:**

Hello, Alice!

The sum is: 8

In this example:

* The mymodule.py file is imported using the import statement.
* You can access the functions in the module using mymodule.function\_name().

**Importing specific functions from a module**

You can also import specific functions directly from a module using the from keyword, which allows you to avoid using the module name every time.

from mymodule import greet, add

greet("Bob")

result = add(10, 4)

print(f"The sum is: {result}")

**Output:**

Hello, Bob!

The sum is: 14

**Renaming a module during import**

You can use the as keyword to rename the module while importing it, which is useful for shortening long module names.

import mymodule as mm

mm.greet("Charlie")

result = mm.add(3, 7)

print(f"The sum is: {result}")

**Output:**

Hello, Charlie!

The sum is: 10

**3. Built-in Python Modules**

Python comes with a set of built-in modules, which you can use directly in your code without needing to install anything. Some examples include:

* **math**: Provides mathematical functions such as sqrt(), pow(), sin(), etc.
* **random**: Used to generate random numbers.
* **sys**: Provides access to system-specific parameters and functions.
* **datetime**: Used to work with dates and times.

**Example: Using the math module**

import math

print(f"The square root of 16 is: {math.sqrt(16)}")

print(f"The value of pi is: {math.pi}")

**Output:**

The square root of 16 is: 4.0

The value of pi is: 3.141592653589793

RANDOM AND MATHS MODULE

**1. math Module**

The math module provides mathematical functions to perform basic and advanced mathematical operations. This module is part of Python’s standard library, so no installation is required.

**Common Functions in math Module:**

1. **math.sqrt(x)**: Returns the square root of x.

import math

print(math.sqrt(16)) # Output: 4.0

1. **math.pow(x, y)**: Returns x raised to the power of y.

print(math.pow(2, 3)) # Output: 8.0

1. **math.pi**: Constant representing the value of pi (3.141592653589793).

print(math.pi) # Output: 3.141592653589793

1. **math.ceil(x)**: Returns the smallest integer greater than or equal to x.

print(math.ceil(4.3)) # Output: 5

1. **math.floor(x)**: Returns the largest integer less than or equal to x.

print(math.floor(4.7)) # Output: 4

1. **math.factorial(x)**: Returns the factorial of x (x!).

print(math.factorial(5)) # Output: 120

1. **math.gcd(a, b)**: Returns the greatest common divisor (GCD) of a and b.

print(math.gcd(24, 36)) # Output: 12

1. **math.radians(x)**: Converts degrees to radians.

print(math.radians(90)) # Output: 1.5707963267948966

1. **math.degrees(x)**: Converts radians to degrees.

print(math.degrees(math.pi)) # Output: 180.0

1. **math.log(x, base)**: Returns the logarithm of x to the given base.

print(math.log(100, 10)) # Output: 2.0

**2. random Module**

The random module provides functions that allow you to generate random numbers, make random choices, shuffle elements, and more. It’s especially useful for simulations, games, or any situation where randomness is needed.

**Common Functions in random Module:**

1. **random.random()**: Returns a random floating-point number between 0 and 1.

import random

print(random.random()) # Output: A random number between 0 and 1

1. **random.randint(a, b)**: Returns a random integer between a and b (inclusive).

print(random.randint(1, 10)) # Output: A random integer between 1 and 10

1. **random.uniform(a, b)**: Returns a random floating-point number between a and b.

print(random.uniform(5, 10)) # Output: A random number between 5 and 10

1. **random.choice(sequence)**: Returns a random element from a non-empty sequence (e.g., list, tuple).

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

print(random.choice(fruits)) # Output: A random element from the list

1. **random.shuffle(sequence)**: Randomly shuffles the elements of a list in place.

items = [1, 2, 3, 4, 5]

random.shuffle(items)

print(items) # Output: The list is shuffled randomly

1. **random.sample(sequence, k)**: Returns a list of k random elements chosen from the sequence without replacement.

print(random.sample(fruits, 2)) # Output: A random sample of 2 elements from the list

1. **random.seed(a)**: Initializes the random number generator with a seed value a. This is useful for reproducibility.

random.seed(42)

print(random.randint(1, 100)) # Output: Same random number every time with the same seed

1. **random.gauss(mu, sigma)**: Returns a random number drawn from a Gaussian distribution with mean mu and standard deviation sigma.

print(random.gauss(0, 1)) # Output: A random number from a normal distribution

**CREATING A CUSTOM MODULE**

Creating custom modules in Python is a great way to organize your code and make it reusable

**Create a Python File**: Start by creating a new Python file (e.g., mymodule.py). This file will contain the functions, classes, and variables you want to include in your module.

**Define Functions and Classes**: Write the functions and classes you want to include in your module. For example:

python

# mymodule.py

def greet(name):

return f"Hello, {name}!"

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def introduce(self):

return f"My name is {self.name} and I am {self.age} years old."

**Save the Module**: Save the file with a .py extension. This file is now your custom module.

**Import the Module**: To use your custom module in another Python script, use the import statement. For example:

python

# main.py

import mymodule

print(mymodule.greet("Alice"))

person = mymodule.Person("Bob", 30)

print(person.introduce())

**Run Your Script**: Run your main script (e.g., main.py) to see the output from your custom module