Function,Modules by Mrittika Megaraj

Function

```
A function is a block of code that performs a specific task.
Code reusability : I don't need to write same code again and again
Pass values to the function : Parameters
It will a value using a keyword called as return
```

In [1]:
```
# user-defined function
def hello():
    print('Hi World')
hello()    #function calling
```

```
Hi World
```

In [2]:
```
# Function along parameter
def addition(n1,n2):
    print(n1+n2)
addition(50,50)
```

```
100
```

Function return type

In [3]:
```
# function
def find_square(num):
    result = num * num
    return result
#calling function
square = find_square(10)

print('Square:',square)
```

```
Square: 100
```

Types of Function Arguments:

1. Default Arguments:

In [4]:
```
def add_numbers( a = 70,  b = 80):
    sum = a + b
    print('Sum:', sum)

# two arguments
add_numbers(2, 3)

#No arguments
add_numbers()
```

```
Sum: 5
Sum: 150
```

2. Keyword Arguments

In [5]:
```
def show(first_name, last_name):
    print('First Name:', first_name)
    print('Last Name:', last_name)

show(last_name = 'Megaraj', first_name = 'Mrittika')
```

```
First Name: Mrittika
Last Name: Megaraj
```

3. Positional Arguments

In [6]:
```
def prints(age,name):
    print(age,name)

prints('Mrittika',21)
prints(21,'Mrittika')
```

```
Mrittika 21
21 Mrittika
```

4. Arbitrary Arguments

In [7]:
```
#find sum of multiple numbers
def find_sum(*numbers):
    result = 0
    for num in numbers:
        result = result + num
    print("Sum = ", result)

find_sum(1, 2, 3)
```

```
Sum =  6
```

Python Recursion : Recursion is the process of defining something in terms of itself.

In [8]:
```
def factorial(x):
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))

x = int(input("Enter the number:"))
print("The factorial is", factorial(x))
```

```
Enter the number:6
The factorial is 720
```

In [9]:
```
#python recursive pattern
def row(n):
    if n < 1:
        return
    print("*", end=" ")
    row(n - 1)

def pattern(n):
    if n < 1:
        return
    row(n)
    print("")
    pattern(n - 1)

n = 5
pattern(n)
```

```
* * * * *
* * * *
* * *
* *
*
```

Modules

```
There are two types of modules:
1.Built-in module:
2.User-defined module:
```

MATH Function

In [10]:
```python
import math

# Calculate the square root of a number
num = 25
sqrt_num = math.sqrt(num)
print(f"The square root of {num} is {sqrt_num}")

# Calculate the factorial of a number
num = 5
factorial = math.factorial(num)
print(f"The factorial of {num} is {factorial}")

# Calculate the value of pi
pi_value = math.pi
print(f"The value of pi is approximately {pi_value}")

# Calculate the sine of an angle in radians
angle_rad = math.radians(30)  # Convert 30 degrees to radians
sin_value = math.sin(angle_rad)
print(f"The sine of 30 degrees is {sin_value}")

# Calculate the natural logarithm (base e) of a number
num = 2.71828  # Euler's number (approximately)
ln_value = math.log(num)
print(f"The natural logarithm of {num} is {ln_value}")

# Calculate the power of a number
base = 2
exponent = 3
power_result = math.pow(base, exponent)
print(f"{base} raised to the power of {exponent} is {power_result}")

# Round a number to the nearest integer
num = 3.7
rounded_num = math.ceil(num)  # ceil() rounds up, floor() rounds down
print(f"{num} rounded to the nearest integer is {rounded_num}")
```

```
The square root of 25 is 5.0
The factorial of 5 is 120
The value of pi is approximately 3.141592653589793
The sine of 30 degrees is 0.49999999999999994
The natural logarithm of 2.71828 is 0.999999327347282
2 raised to the power of 3 is 8.0
3.7 rounded to the nearest integer is 4
```

DateTime Function

In [11]:
```python
import datetime

# Get the current date and time
current_datetime = datetime.datetime.now()
print(f"Current Date and Time: {current_datetime}")

# Get the current date
current_date = datetime.date.today()
print(f"Current Date: {current_date}")

# Create a specific date
specific_date = datetime.date(2023, 9, 30)
print(f"Specific Date: {specific_date}")

# Create a specific time
specific_time = datetime.time(14, 30, 0)
print(f"Specific Time: {specific_time}")

# Combine date and time into a datetime object
combined_datetime = datetime.datetime.combine(specific_date, specific_time)
print(f"Combined DateTime: {combined_datetime}")

# Access individual components of a datetime object
year = current_datetime.year
month = current_datetime.month
day = current_datetime.day
hour = current_datetime.hour
minute = current_datetime.minute
second = current_datetime.second

print(f"Year: {year}")
print(f"Month: {month}")
print(f"Day: {day}")
print(f"Hour: {hour}")
print(f"Minute: {minute}")
print(f"Second: {second}")
```

```
Current Date and Time: 2024-02-02 21:34:43.652612
Current Date: 2024-02-02
Specific Date: 2023-09-30
Specific Time: 14:30:00
Combined DateTime: 2023-09-30 14:30:00
Year: 2024
Month: 2
Day: 2
Hour: 21
Minute: 34
Second: 43
```

User Defined Function

In [12]:
```
num1 = int(input("Enter first number:"))

num2 = int(input("Enter second number:"))

print("Addition", num1 + num2)

print("Subtraction", num1 - num2)
print("Multiplication", num * num2)
print("Division",num1/num2)
```

```
Enter first number:10
Enter second number:2
Addition 12
Subtraction 8
Multiplication 7.4
Division 5.0
```

In [13]:
```
import random

# Generate a random integer between 1 and 10 (inclusive)
random_num = random.randint(1, 10)
print("Random Number:", random_num)

# Generate a random floating-point number between 0 and 1
random_float = random.random()
print("Random Float:", random_float)
```

```
Random Number: 8
Random Float: 0.8587508337194668
```

In [14]:
```
import json

# Serialize Python dictionary to JSON
data = {"name": "Alice", "age": 30}
json_data = json.dumps(data)
print("JSON Data:", json_data)

# Deserialize JSON to Python dictionary
parsed_data = json.loads(json_data)
print("Python Dictionary:", parsed_data)
```

```
JSON Data: {"name": "Alice", "age": 30}
Python Dictionary: {'name': 'Alice', 'age': 30}
```

Variable Scope

In [15]:
```
n,m = 10,20
print(n,m)

def func(n):
    a = 10
    global b
    b=12
    print(a,n,b)
    print(m)

func(n)
print(n,m)
```

```
10 20
10 10 12
20
10 20
```

yield vs return

In [18]:
```
# return

def fun1(n):
    for i in range(2,n):
        return i**2

print(fun1(5))
```

```
4
```

In [19]:
```
# yield

def fun2(n):
    for i in range(1,n):
        yield i**2

print(list(fun2(5)))
```

```
[1, 4, 9, 16]
```

In [ ]:
```

```