PYTHON

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language.

Characteristics of Python

Following are important characteristics of Python Programming –

- *It supports functional and structured programming methods as well as OOP.*
- *It can be used as a scripting language or can be compiled to byte-code for building large applications.*
- *It provides very high-level dynamic data types and supports dynamic type checking.*
- *It supports automatic garbage collection.*
- *It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.*

Advantages:

- *Presence of third-party modules*
- *Extensive support libraries(NumPy for numerical calculations, Pandas for data analytics, etc.)*
- *Open source and large active community base*
- *Versatile, Easy to read, learn and write*
- *User-friendly data structures*
- *High-level language*
- *Dynamically typed language(No need to mention data type based on the value assigned, it takes data type)*
- *Object-Oriented and Procedural Programming language*
- *Portable and Interactive*
- *Ideal for prototypes – provide more functionality with less coding*

Operations using PYTHON :

In [1]:
```
print("hello world!!")
```

```
hello world!!
```

In [6]:
```
x =12;
if(x<10):
    print("I am lesser than 10")
else:
        print("I am greater than 10")
```

```
I am greater than 10
```

In [9]:
```
for x in range(1,10):
    print(x+1)
```

```
2
3
4
5
6
7
8
9
10
```

In [40]:
```
mystr = "welcome to home !!!"
print(mystr.capitalize())
print(mystr.casefold())
print(mystr.center(5))
print(mystr.isupper())
print(mystr.islower())
print(mystr.find("e"))
print (mystr.splitlines(7))
```

```
Welcome to home !!!
welcome to home !!!
welcome to home !!!
False
True
1
['welcome to home !!!']
```

In [41]:
```
my_tuple = (10, 20, 30)
print(my_tuple[0])  # Output: 10
print(my_tuple[-1])  # Output: 30
```

```
10
30
```

In [42]:
```
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple[1:4])  # Output: (2, 3, 4)
```

```
(2, 3, 4)
```

In [43]:
```
my_tuple = ('apple', 'banana', 'cherry')
for fruit in my_tuple:
    print(fruit)
```

```
apple
banana
cherry
```

In [44]:
```
my_tuple = (1, 2, 2, 3, 2, 4)
print(my_tuple.count(2))
```

```
3
```

In [45]:
```
my_tuple = ('apple', 'banana', 'cherry')
print(my_tuple.index('banana'))
```

```
1
```

In [46]:
```
my_tuple = (10, 20, 30)
print(20 in my_tuple)
print(40 in my_tuple)
```

```
True
False
```

In [47]:
```
my_tuple = (1, 2)
repeated_tuple = my_tuple * 3
print(repeated_tuple)
```

```
(1, 2, 1, 2, 1, 2)
```

In [48]:
```
tuple1 = (1, 2, 3)
tuple2 = ('a', 'b', 'c')
concatenated_tuple = tuple1 + tuple2
print(concatenated_tuple)
```

```
(1, 2, 3, 'a', 'b', 'c')
```

In [49]:
```
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple[1:4])
```

```
(2, 3, 4)
```

In [50]:
```
my_tuple = (10, 5, 15, 20, 8)
print(min(my_tuple))
print(max(my_tuple))
```

```
5
20
```

In [51]:
```
person_info = ('John', 30, 'Male')
name, age, gender = person_info
print(name)
print(age)
print(gender)
```

```
John
30
Male
```

In [52]:
```
my_list = [1, 2, 3, 4]
tuple_from_list = tuple(my_list)
print(tuple_from_list)
```

```
(1, 2, 3, 4)
```

In [54]:
```
person_info = ['John', 30, 'Male']
name, age, gender = person_info
print(age)
print(name)
print(gender)
```

```
30
John
Male
```

In [55]:
```
my_list = [10, 5, 15, 20, 8]
print(min(my_tuple))
print(max(my_tuple))
```

```
5
20
```

In [56]:
```
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
concatenated_list = list1 + list2
print(concatenated_list)
```

```
[1, 2, 3, 'a', 'b', 'c']
```

In [57]:
```
my_list = [1, 2, 3, 4, 5]
print(my_list[1:4])
```

```
[2, 3, 4]
```

In [58]:
```
my_list = [1, 2, 3, 4, 5]
print(len(my_list))
```

```
5
```

In [59]:
```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)
```

```
[1, 2, 3, 4]
```

In [60]:
```
my_list[-1]
```

Out[60]:
```
4
```

In [61]:
```
my_list = [1, 2, 3, 4, 5]
my_list.remove(3)
print(my_list)

my_list.pop()
print(my_list)
```

```
[1, 2, 4, 5]
[1, 2, 4]
```

In [62]:
```
my_list = ['apple', 'banana', 'cherry']
for fruit in my_list:
    print(fruit)
```

```
apple
banana
cherry
```

In [2]:
```
#Arithmetic Operators:

# Addition
result_add = 10 + 5
print(result_add)  # Output: 15

# Subtraction
result_sub = 20 - 7
print(result_sub)  # Output: 13

# Multiplication
result_mul = 6 * 4
print(result_mul)  # Output: 24

# Division
result_div = 15 / 3
print(result_div)  # Output: 5.0 (Note: division always returns a float in Pyt
hon)

# Modulo (Remainder)
result_mod = 17 % 4
print(result_mod)  # Output: 1

# Exponentiation
result_exp = 2 ** 3
print(result_exp)  # Output: 8

# Floor Division
result_floor_div = 17 // 4
print(result_floor_div)  # Output: 4
```

```
15
13
24
5.0
1
8
4
```

In [3]:
```
#Assignment Operators:

# Assign a value to a variable
x = 10

# Add a value to the variable and assign the result
x += 5
print(x)  # Output: 15

# Subtract a value from the variable and assign the result
x -= 3
print(x)  # Output: 12

# Multiply the variable by a value and assign the result
x *= 2
print(x)  # Output: 24

# Divide the variable by a value and assign the result
x /= 3
print(x)  # Output: 8.0

# Perform modulo operation with the variable and assign the result
x %= 5
print(x)  # Output: 3.0

# Perform exponentiation with the variable and assign the result
x **= 2
print(x)  # Output: 9.0

# Perform floor division with the variable and assign the result
x //= 2
print(x)  # Output: 4.0
```

```
15
12
24
8.0
3.0
9.0
4.0
```

In [4]:
```
#Comparison (Relational) Operators:

# Equal to
print(5 == 5)  # Output: True

# Not equal to
print(10 != 5)  # Output: True

# Greater than
print(8 > 3)  # Output: True

# Less than
print(7 < 2)  # Output: False

# Greater than or equal to
print(6 >= 6)  # Output: True

# Less than or equal to
print(4 <= 2)  # Output: False
```

```
True
True
True
False
True
False
```

In [5]:
```
#Logical Operators:

a = True
b = False

# Logical AND
print(a and b)  # Output: False

# Logical OR
print(a or b)  # Output: True

# Logical NOT
print(not a)  # Output: False
print(not b)  # Output: True
```

```
False
True
False
True
```

In [6]:
```
#Bitwise Operators:

# Bitwise AND
result_and = 0b1100 & 0b1010  # 12 & 10 in binary
print(bin(result_and))  # Output: '0b1000' (8 in decimal)

# Bitwise OR
result_or = 0b1100 | 0b1010  # 12 | 10 in binary
print(bin(result_or))  # Output: '0b1110' (14 in decimal)

# Bitwise XOR
result_xor = 0b1100 ^ 0b1010  # 12 ^ 10 in binary
print(bin(result_xor))  # Output: '0b0110' (6 in decimal)

# Bitwise NOT
result_not = ~0b1100  # ~12 in binary
print(bin(result_not))  # Output: '-0b1101' (-13 in decimal)

# Left shift
result_left_shift = 0b0010 << 2  # 2 shifted left by 2 bits
print(bin(result_left_shift))  # Output: '0b1000' (8 in decimal)

# Right shift
result_right_shift = 0b1010 >> 1  # 10 shifted right by 1 bit
print(bin(result_right_shift))  # Output: '0b0101' (5 in decimal)
```

```
0b1000
0b1110
0b110
-0b1101
0b1000
0b101
```

In [7]:
```
#Membership Operators:

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

# Check if a value is found in the list
print(3 in my_list)  # Output: True

# Check if a value is not found in the list
print(6 not in my_list)  # Output: True
```

```
True
True
```

In [1]:
```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

```
1
2
3
4
5
```

In [2]:
```
message = "Hello, World!"
for char in message:
    print(char)
```

```
H
e
l
l
o
,

W
o
r
l
d
!
```

In [3]:
```
for i in range(1, 6):  # 1 to 5 (6 is exclusive)
    print(i)
```

```
1
2
3
4
5
```

In [4]:
```
for i in range(0, 10, 2):  # 0 to 9 with a step of 2
    print(i)
```

```
0
2
4
6
8
```

In [5]:
```
fruits = ['apple', 'banana', 'cherry']
for index, fruit in enumerate(fruits):
    print(index, fruit)
```

```
0 apple
1 banana
2 cherry
```

In [6]:
```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for row in matrix:
    for element in row:
        print(element)
```

```
1
2
3
4
5
6
7
8
9
```

In [7]:
```
names = ['Alice', 'Bob', 'Charlie']
scores = [85, 92, 78]
for name, score in zip(names, scores):
    print(name, score)
```

```
Alice 85
Bob 92
Charlie 78
```

In [8]:
```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        break
    print(num)
```

```
1
2
```

In [9]:
```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        continue
    print(num)
```

```
1
2
4
5
```

In [10]:
```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
else:
    print("Loop completed without a break.")
```

```
1
2
3
4
5
Loop completed without a break.
```

In [1]:
```
input("Enter the number")
```

```
Enter the number3
```

Out[1]:
```
'3'
```

In [2]:
```
num1, num2=input("Enter the two input values").split(' ')
print(int(num1)+int(num2))
```

```
Enter the two input values3 4
7
```

In [5]:
```
# Reading from a file
with open('C:\\Users\\Administrator\\Desktop\\santhu.txt', 'r') as file:
    # Method 1: read the entire file content as a single string
    content = file.read()
    print(content)
```

```
HMS_using_REACT.JS

A Simple Hospital Management System using React.js with EFCORE web API , JWT
Authorization and Authentication !!!!
1860 500 9900
```

In [9]:
```
# Writing to a file
with open('C:\\Users\\Administrator\\Desktop\\santhu.txt', 'w+') as file:
    file.write("This is the first line.\n")
    file.write("This is the second line.\n")
    file.write("This is the third line.\n")
    content = file.read()
    print(content)
```

In [10]:
```
with open('C:\\Users\\Administrator\\Desktop\\santhu.txt', 'w+') as file:
    file.write("This is the first line.\n")
    file.write("This is the second line.\n")
    file.write("This is the third line.\n")

    # Reposition the file pointer to the beginning of the file
    file.seek(0)

    # Read the content after writing
    content = file.read()
    print(content)
```

```
This is the first line.
This is the second line.
This is the third line.
```

In [16]:
```
# Appending to a file
with open('C:\\Users\\Administrator\\Desktop\\santhu.txt', 'w') as file:
    file.write("This is a new line appended to the file.\n")
with open('C:\\Users\\Administrator\\Desktop\\santhu.txt', 'r') as file:
    content = file.read()
    print(content)
```

```
This is a new line appended to the file.
```

In [19]:
```
def mfun1():
    print("this is function")
```

In [20]:
```
mfun1()
```

```
this is function
```

In [21]:
```
def greet(name):
    return "Hello, " + name + "!"

message = greet("John")
print(message)  # Output: Hello, John!
```

```
Hello, John!
```

In [22]:
```
def add_numbers(a, b=0):
    return a + b

result1 = add_numbers(5, 3)
print(result1)  # Output: 8

result2 = add_numbers(10)
print(result2)  # Output: 10 (default value for 'b' is used)
```

```
8
10
```

In [23]:
```
def get_square_and_cube(number):
    square = number ** 2
    cube = number ** 3
    return square, cube

result_square, result_cube = get_square_and_cube(4)
print(result_square)  # Output: 16
print(result_cube)    # Output: 64
```

```
16
64
```

In [24]:
```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

result = factorial(5)
print(result)  # Output: 120 (5! = 5 * 4 * 3 * 2 * 1)
```

```
120
```

In [25]:
```
def average(*args):
    return sum(args) / len(args)

result1 = average(2, 4, 6)
print(result1)  # Output: 4.0

result2 = average(1, 3, 5, 7, 9)
print(result2)  # Output: 5.0
```

```
4.0
5.0
```

In [26]:
```
square = lambda x: x ** 2

result = square(5)
print(result)  # Output: 25
```

```
25
```

In [27]:
```
def print_person_info(name, age, city):
    print("Name:", name)
    print("Age:", age)
    print("City:", city)

print_person_info(name="John", age=30, city="New York")
```

```
Name: John
Age: 30
City: New York
```

In [28]:
```
def calculate_square(num):
    return num ** 2

def calculate_sum_of_squares(a, b):
    square_a = calculate_square(a)
    square_b = calculate_square(b)
    return square_a + square_b

result = calculate_sum_of_squares(3, 4)
print(result)  # Output: 25 (3^2 + 4^2 = 9 + 16 = 25)
```

```
25
```

In [2]:
```
class Employee:
    def work(self):
        return("employee working")


chet = Employee()
alwin = Employee()
print(alwin.work())
```

```
employee working
```

In [ ]:

In [ ]: