PYTHON ASSIGNMENT

1 . Explain the key features of python that make it a popular choice for programing.

Ans :

- **Easy to Read and Write-** Python's syntax is straightforward and easy to understand, which makes it a great choice for both beginners and experienced programmers.
- **Comprehensive Standard Library-** Python's syntax is straightforward and easy to understand, which makes it a great choice for both beginners and experienced programmers.
- **Versatility-** Python is incredibly versatile and can be used for various applications, including web development, data analysis, scientific computing, artificial intelligence, and more.
- **Interpreted Language-** As an interpreted language, Python runs code line by line, making debugging easier and development faster since you don't need to compile the code before running it.
- **Cross-Platform Compatibility -** Python has a vast and active community that provides support, tutorials, and extensive documentation.

2. Describe the role or predefined keywords in python and provide examples of how they are used in a program

Ans : Predefined keywords in Python are reserved words that have special meanings and are an integral part of the language's syntax. They cannot be used as identifiers (names for variables, functions, classes, etc.). These keywords help define the structure and behavior of Python programs.

Predefine functions = if ,elif , else , for ,while , break , continue , def , return, class , self , try , except , finally , import , form , as ,with , lambda, global , nonlocal , true , false , none

Examples 1

```
    x = 10
if x > 0:
  print("Positive")
elif x == 0:
  print("Zero")
else:
  print("Negative")
```

Example 2

```
# for loop
```

```
for i in range(5):
   if i == 3:
      break  # Exit the loop
   print(i)


# while loop
i = 0
while i < 5:
   i += 1
   if i == 3:
      continue  # Skip the rest of the loop body for this iteration
   print(i)
```

Example 3

```
a = True
b = False
c = None


if a:
   print("a is True")
if not b:
   print("b is False")
if c is None:
   print("c is None")
```

3. Compare and contrast mutable and immutable objects in python with example

Ans - In Python, objects can be classified as either mutable or immutable based on whether their state (data) can be changed after they are created. Here's a detailed comparison

Mutable Objects

- **Definition**: Mutable objects are those whose state or data can be changed after they are created.
- **Examples**: Lists, dictionaries, sets, byte arrays.
- **Behavior**: Modifications to mutable objects are done in place, meaning that the object's identity remains the same even though its contents may change.

Example

```
# Create a list (mutable)

my_list = [1, 2, 3]


# Modify the list

my_list[0] = 4


print(my_list)  # Output: [4, 2, 3]


# The list's identity remains the same

print(id(my_list))

# Output: The same memory address as before
```

Immutable

- **Definition**: Immutable objects are those whose state or data cannot be changed after they are created.
- **Examples**: Strings, tuples, integers, floats, frozensets, bytes.
- **Behavior**: Any operation that appears to modify an immutable object will instead create a new object with the modified value, leaving the original object unchanged.

Example

```
 # Create a string (immutable)

my_string = "hello"


# Attempt to modify the string

new_string = my_string.replace("h", "j")
```

```
print(my_string)  # Output: 'hello'

print(new_string)  # Output: 'jello'


# The string's identity remains the same for the original string

print(id(my_string))  # Output: Original memory address

print(id(new_string))  # Output: New memory address
```

4. Discuss the different types of operators in python and provide examples of how they are used.

Ans : Python has several types of operators that allow you to perform various operations on variables and values. Here's a detailed overview of the different types of operators and examples of how they are used

## operators with examples

i. Arithmetic Operators

These operators perform mathematical operations.

- Addition (+): a = 5; b = 3; result = a + b (result is 8)
- Subtraction (-): result = a - b (result is 2)
- Multiplication (*): result = a * b (result is 15)
- Division (/): result = a / b (result is 1.6666666666666667)
- Floor Division (//): result = a // b (result is 1)
- Modulus (%): result = a % b (result is 2)
- Exponentiation (**): result = a ** b (result is 125)

ii. Comparison Operators

These operators compare two values and return a boolean result.

- Equal (==): result = (a == b) (result is False)
- Not Equal (!=): result = (a != b) (result is True)
- Greater Than (>): result = (a > b) (result is True)
- Less Than (<): result = (a < b) (result is False)

- Greater Than or Equal To (>=): result = (a >= b) (result is True)
- Less Than or Equal To (<=): result = (a <= b) (result is False)

iii. Logical Operators

These operators are used to combine conditional statements.

- Logical AND (and): result = (a > 0 and b > 0) (result is True)
- Logical OR (or): result = (a > 0 or b < 0) (result is True)
- Logical NOT (not): result = not (a > b) (result is True)

iv. Bitwise Operators

These operators perform bit-level operations on binary numbers.

- Bitwise AND (&): result = a & b (result is 1)
- Bitwise OR (|): result = a | b (result is 7)
- Bitwise XOR (^): result = a ^ b (result is 6)
- Bitwise NOT (~): result = ~a (result is -6)
- Left Shift (<<): result = a << 1 (result is 10)
- Right Shift (>>): result = a >> 1 (result is 2)

v. Assignment Operators

These operators are used to assign values to variables.

- Simple Assignment (=): a = 10
- Add and Assign (+=): a += 5 (now a is 15)
- Subtract and Assign (-=): a -= 3 (now a is 12)
- Multiply and Assign (*=): a *= 2 (now a is 24)
- Divide and Assign (/=): a /= 4 (now a is 6.0)
- Floor Divide and Assign (//=): a //= 2 (now a is 3.0)
- Modulus and Assign (%=): a %= 2 (now a is 1.0)
- Exponentiate and Assign (**=): a **= 3 (now a is 1.0)

vi. Identity Operators

These operators compare the memory locations of two objects.

- Is (is): result = (a is b) (True if a and b are the same object)
- Is Not (is not): result = (a is not b) (True if a and b are not the same object)

vii. Membership Operators

These operators test for membership in a sequence (such as strings, lists, or tuples).

- In (in): my_list = [1, 2, 3, 4, 5]; result = (3 in my_list) (result is True)
- Not In (not in): result = (6 not in my_list) (result is True)

viii. Conditional (Ternary) Operator

This operator evaluates a condition and returns one of two values based on the condition.

x = 5; y = 10; result = "x is greater" if x > y else "y is greater" (result is "y is

5. Explain the concept of type casting in python with example.

Ans:

Type casting in Python refers to the conversion of one data type into another. This can be done using Python's built-in functions. Type casting is useful when you need to perform operations that require operands of a specific data type.

1. Implicit Type Casting

In implicit type casting, Python automatically converts one data type to another. This usually happens when different types of data are involved in an expression, and Python converts the data type to avoid data loss.

Example:

```
# Implicit type casting
a = 10   # Integer
b = 3.5  # Float

# Python automatically converts the integer to float before addition
result = a + b

print(result)       # Output: 13.5
print(type(result))   # Output: <class 'float'>
```

#. Explicit Type Casting

In explicit type casting, also known as type conversion, you manually convert one data type to another using Python's built-in functions.

Common functions used for explicit type casting:

- int(): Converts a value to an integer
- float(): Converts a value to a float
- str(): Converts a value to a string
- list(): Converts a value to a list
- tuple(): Converts a value to a tuple
- set(): Converts a value to a set

Examples:

- Integer to Float:

```
a = 10  # Integer
b = float(a)  # Convert integer to float
print(b)       # Output: 10.0
print(type(b))   # Output: <class 'float'>
```

6. How do conditional statements work In python with example.
Ans :
Conditional statements in Python are used to execute specific blocks of code based on whether a condition is `True` or `False`. The primary conditional statements in Python are `if`, `elif`, and `else`.

Example If

```
# Check if a number is positive
        number = 10 if number > 0:
        print("The number is positive.")
# Output: The number is positive.
```

Example if-else

# Check if a number is positive or negative number = -5

if number > 0: print("The number is positive.")

else: print("The number is negative or zero.")

# Output: The number is negative or zero.

Example if-elif-else

# Check if a number is positive, negative, or zero

number = 0

if number > 0:

print("The number is positive.")

elif number < 0: print("The number is negative.")

else: print("The number is zero.")

# Output: The number is zero.

7. Describe the different types of loops in python and there use case with example

Ans :

Python provides two primary types of loops: `for` loops and `while` loops. These loops are used to execute a block of code repeatedly, depending on the specified conditions.

Example for

# List of numbers

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

# Using a for loop to iterate over the list and print each number for number in numbers:

print(number)

# output

1

2

3

4

Example while

```
# Initialize a variable
count = 1

# Using a while loop to print numbers from 1 to 5
while count <= 5:
   print(count)
   count += 1
```

Output:

```

2
3
4
5
```