Assignment - 3

1. Explain Slicing in Strings with different Examples.

⇒ Slicing Strings:

→ To return a range of characters by using the slice syntax.

→ Specify the start index and the end index separated by a colon, to return a part of the string.

→ Example : Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Output: llo

Note : The first character has index 0

⟹ Slice from the start:

→ By leaving out the start index, the range will start at the first character

→ Example : Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

Output:

Hello

⟹ Slice to the End

→ By leaving out the end index, the : range will go to the end:

→ Example: [Get the characters from position 2, and all the way to the end:]

```
b = "Hello, World!"
print(b[2:])
```

Output:

llo, World!

⟹ Negative Indexing

→ Use negative indexes to start the slice from the end of the string:

→ Example : Get the characters:

From : "o" in "World!" (-5) to, "d" in "World!" (-2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Output:

orl

2. Short note on String Method.

→ Basically there are 10 method for string

→ 1. center() :-

This will center align the string using a specified character as the fill character

→ Eg : txt = "banana"

```
x = txt.center(20)
print(x)
```

output

banana

→ 2. Count() :-

This will return number of times specified value appears inthe string.

→ Eg txt = "A b b , b c"

```
x = txt.count('b')
print(x)
```

output

3

→ 3. Join() :-

Method takes all items in an iterable and joins them into one string. A string must be specified as the seperator

→ Eg: txt = ("A","B")

```
x = "#".join(txt)
print(x)
```

output

A#B

→ 4. len() :-

To get the length of string.

Eg: a = "Hello"

```
x = len(a)
print(x)
```

output

5

→ 5. max():-

It will return largest character in a string

Eg: txt = "ABXZ"
x = max(txt)
print(x)

Output
Z

→ 6. min():-

To get smallest character from string

Eg: a = "ABXZ"
print(min(a))

Output
A

→ 7. replace:-

Method replaces a specified phrase with another specified phrase. By default, All occurrences of the specified phrase will be replaced.

Ex txt = " one one banana"
x = txt.replace("one", "ok")
print(x)

Output
ok ok banana

→ 8. lower:-

Method returns a string where all characters are lower case. Symbols and numbers are ignored

Ex. txt = " Hello "
x = txt.lower()
print(x)

Output
hello

→ 9. upper :-

The upper() method returns a string where all characters are in upper case. Symbols and Numbers are ignored.

Ex : txt = "hello"
x = txt.upper()
print(x)

Output
HELLO

→ 10. split :-

The split() method splits a string into a list. You can specify the separator, default separator is any whitespace.

Ex : txt = "Welcome to Jungle"
x = txt.split()
print(x)

→ Output
['welcome', 'to', 'Jungle']

3. Explain Operators in Python in Detail.

→ Operators are used to perform operations on variables and values.

→ Python divides the operators in the following groups:

1. Arithmetic operators
2. Comparison operators
3. Assignment operators
4. Logical operators
5. Identity and member operator

1. Python Arithmetic Operators

→ Arithmetic operators are used with numeric values to perform common mathematical operations:

(i) Addition operator
Operator = +
Example = x + y

(ii) Subtraction operator to subtract from operand
operator = -
Example = x - y

(iii) Multiplication operator
Operator = *
Exampl = x * y

(iv) Division Operator
Operator = /
Example = x / y

(v) Modulus Operator to get remainder
Operator = %
Example = x % y

(vi) Exponentiation operator
Operator = **
Example = x ** y

(vii) Floor division Operator
Operator = //
Example = x // y

2. Python Comparison Operators
→ Comparison operators are used to compare two values.
→ (i) is Equal operator ==
Example : x == y

(ii) is NOT Equal !=
Example : x != y

(iii) > Greater than >
Example : x > y

(iv) Less than <
Example : x < y

(v) Greater than or equal to >=
Example : x >= y

(vi) Less than or equal to <=
Example : x <= y

3. Python Assignment Operators
→ Assignment operators are used to assign values to variables.

(i) Operater = use to assign value
Example x = 5

(ii) += use to add variable value + value
Example x += 5 mean x = x + 5

(iii) -= use to subtract variable value - value
Example x -= 3 mean x = x - 3

(iv) *= use for multiplication then assign
Example x *= 4 same as x = x * 4

(v) /= use for division then assign
Example x /= 4 same as x = x / 4

(vi) %= use for get remainder then assign
Example x %= 2 same as = x = x % 2

(vii) //= use for fbor division then assign
Example x //= 2 same as x = x // 2

(viii) **= use for exponentiation then assign
Example x **= 3 same as x = x ** 3

4. Python Logical Operators
→ Logical operators are used to combine conditional statements
→ 1. and
→ and operators return true if both statements are true (x=1) output
→ Example x < 5 and x < 10 1 (True)

→ 2. Or
→ return true if any one statements is true
Example 5 == x or x > 4 0 (False)

→ 3. Not
→ Reverse the result
Example: not (0 > 1) 1 (True)

→ 5. Pyth

5. Python identity Operators
→ identity operators are used to compare the objects, not if they are equal but if they are actually the same object, with same memory location.
→ 1. Is
→ Returns true if both variables are same
Exa: x is y → 0 (False)

→ 2. Is not
→ Returns True if both variables are not same
→ Ex: x is not y → 1 (True)

2. Member Operator:

→ Membership operators are used to test if a sequence is presented in an object:

→ 1. in

→ Returns true if a sequence with the specified value is present in the object

→ Example: x = ("apple", "banana")
print("appple" in x)

Output: True.

→ 2. Not in

→ Returns true if a sequence with the specified value is not present in the object

→ Ex: x = ("apple", "car")
print("bike" not in x)

Output: True