**Python training - Session 2**

**Conditional Statement:**

**If statement:** This condition used for the whether the condition is true or false or satisfy or not**.**

**Example:**

a = 33

b = 200

if b > a:

print("b is greater than a")

**Else statement:** The else keyword catches anything which isn't caught by the preceding conditions.

Example:

a = 33

b = 200

if b > a:

print("b is greater than a")

else:

print(‘a is greater than b’)

**Elif statement:** The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

Example:

a = 33

b = 33

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

**Break :** It is used for the termination.

Example:

for x in range(1,10):

if x==5:# if x is equal to 5 then stop the iteration and exit to this condition

break

**Continue:** Continue statement is used for the skip the current iteration and start to next iteration.

Example:

For i in range(1,10):

If i==5:

# if I equals to 5 that iteration skip next enter into next iteration

Continue

**Pass:** It means No operation

**Looping statement:**

**For loop:** A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Example:

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

for x in fruits:

print(x)

Example 2:

for x in "banana":

print(x)

**While loop:**

With the while loop we can execute a set of statements as long as a condition is true.

Example:

i = 1

while i < 6:

print(i)

i += 1

Example 1:

i = 1

while i < 6:

print(i)

if i == 3:

break

i += 1

**Operators in Python:**

* A**rithmetic operator:**

Arithmetic operators are used with numeric values to perform common mathematical operations. +, -, \*, / these are all the arithmetic operators.

Example:

a=10

b=20

print(a+b)

print(a-b)

print(a/b)

* **Assignment operator:**

Assignment operators are used to assign values to variables.

Example:

X=5, x+=10, x-=90, x/=10, x\*=10

These type are all assignment operators.

* **Comparison operator:**

Comparison operators are used to compare two values.

Comparison operators are <, >, <=, >=, =!, ==,

Example:

A=10

B=10

If a==b:

Print(‘both are equal’)

Example 2:

A=10

B=20

if a<b:

print(‘ a is smaller than b’)

* **Logical operator:**

Logical operators are used to combine conditional statements.

The logical operator are,

**And:** Returns True if both statements are true.

Example:

if a>10 and b<10:

Return True

**Or:** Returns True if any one of the statement is true.

Example:

If a<10 or b>20:

Return true

**Not:** Reverse the result, returns False if the result is true.

Example: not(x < 5 and x < 10)

* **Identity operators:** Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

The two operators are there is and is not,

**Is:** Returns True if both variables are the same object.

Example:

X=10

Y=10

If x is y:

Return True

**Is not:** Returns True if both variables are not the same object.

Example:

X=10

Y=20

If x is not y:

Return True

* **Membership operator:**

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

The two membership operator are there they are,

In and not in

**In:** Returns True if a sequence with the specified value is present in the object.

Example: x in y

**Not in:** Returns True if a sequence with the specified value is not present in the object.

Example: x not in y

* **Bitwise operators:**

Bitwise operators are used to compare (binary) numbers.

The bitwise operators they are given below,

**And(&):** Sets each bit to 1 if both bits are 1.

**Or(|):** Sets each bit to 1 if one of two bits is 1.

**Xor(^):** Sets each bit to 1 if only one of two bits is 1.

**Not(~):** Inverts all the bits.

**Zero fill left shift(<<):** Shift left by pushing zeros in from the right and let the leftmost bits fall off.

**Signed right shift(>>):** Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off.