**Loops In Python**



Python programming language provides following types of loops to handle looping requirements.

|  |  |
| --- | --- |
| **Sr.No.** | **Loop Type & Description** |
| 1 | [while loop](https://www.tutorialspoint.com/python/python_while_loop.htm)  Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. |
| 2 | [for loop](https://www.tutorialspoint.com/python/python_for_loop.htm)  Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. |
| 3 | [nested loops](https://www.tutorialspoint.com/python/python_nested_loops.htm)  You can use one or more loop inside any another while, for or do..while loop. |

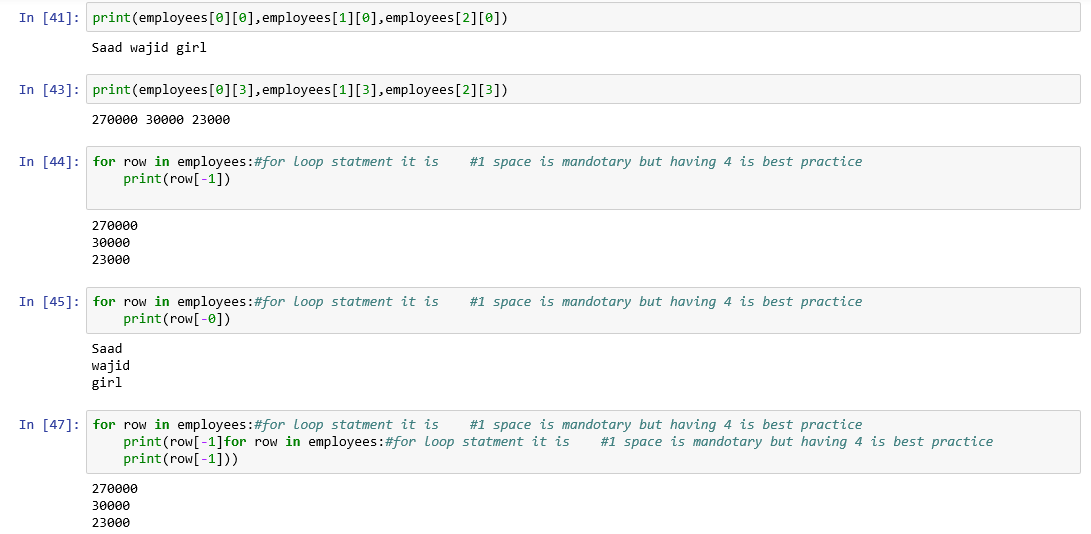
Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements. Click the following links to check their detail.

Let us go through the loop control statements briefly

|  |  |
| --- | --- |
| **Sr.No.** | **Control Statement & Description** |
| 1 | [break statement](https://www.tutorialspoint.com/python/python_break_statement.htm)  Terminates the loop statement and transfers execution to the statement immediately following the loop. |
| 2 | [continue statement](https://www.tutorialspoint.com/python/python_continue_statement.htm)  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating. |
| 3 | [pass statement](https://www.tutorialspoint.com/python/python_pass_statement.htm)  The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute. |



**Conditional Statements**

## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

### Example

**If statement:**

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

## Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

If statement, without indentation (will raise an error):

a = 33  
b = 200  
if b > a:  
print("b is greater than a") # you will get an error

## Elif

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

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

## Else

The else keyword catches anything which isn't caught by the preceding conditions.

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

## Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

### Example

**One line if statement:**

if a > b: print("a is greater than b")

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

**One line if else statement:**

a = 2  
b = 330  
print("A") if a > b else print("B")

You can also have multiple else statements on the same line:

### Example

**One line if else statement, with 3 conditions:**

a = 330  
b = 330  
print("A") if a > b else print("=") if a == b else print("B")

## And

The and keyword is a logical operator, and is used to combine conditional statements:

Test if a is greater than b, AND if c is greater than a:

a = 200  
b = 33  
c = 500  
if a > b and c > a:  
  print("Both conditions are True")

## Or

The or keyword is a logical operator, and is used to combine conditional statements:

Test if a is greater than b, OR if a is greater than c:

a = 200  
b = 33  
c = 500  
if a > b or a > c:  
  print("At least one of the conditions is True")

## Nested If

You can have if statements inside if statements, this is called nested if statements.

x = 41  
  
if x > 10:  
  print("Above ten,")  
  if x > 20:  
    print("and also above 20!")  
  else:  
    print("but not above 20.")

## The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

a = 33  
b = 200  
  
if b > a:  
  pass



**TASK:.**

Using conditional statements show the different outcomes of given data.

**TASK:**

Create a for loop to call different attributes of given data.