Python Conditions and If statements

June 2, 2024

0.1 Python Conditions and If statements

**Dr.Labeed Al-Saad

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.

0.2 if statement

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

ExampleGet your own Python Server If statement:

[1]:
```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In the ubove example we used 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.

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

[2]:
```
a = 33
b = 200
if b > a:
```

```
print("b is greater than a") # you will get an error
```

```
  Cell In[2], line 4
    print("b is greater than a") # you will get an error
    ^
IndentationError: expected an indented block after 'if' statement on line 3
```

0.3 Elif

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

Example 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".

[3]:
```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

0.4 Else

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

Example 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".

[4]:
```
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")
```

```
a is greater than b
```

[ ]:
```
You can also have an else without the elif:

Example
```

[5]:
```
a = 200
b = 33
```

```
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

```
b is not greater than a
```

0.5 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:

[6]:
```
if a > b: print("a is greater than b")
```

```
a is greater than b
```

0.6 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:

Example One line if else statement:

[7]:
```
a = 2
b = 330
print("A") if a > b else print("B")
```

```
B
```

**This technique is known as Ternary Operators, or Conditional Expressions.

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

Example One line if else statement, with 3 conditions:

[8]:
```
a = 330
b = 330
print("A") if a > b else print("=") if a == b else print("B")
```

```
=
```

0.7 And

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

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

[9]:
```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

```
Both conditions are True
```

0.8 Or

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

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

[10]:
```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
At least one of the conditions is True
```

0.9 Not

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

Example Test if a is NOT greater than b:

[11]:
```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

```
a is NOT greater than b
```

0.10 Nested If

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

Example

[13]:
```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  if x > 25:
    print("and also above 25!")
  else:
    print("but not above 20.")
```

```
Above ten,
and also above 20!
and also above 25!
```

0.11 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.

Example

[16]:
```
a = 33
b = 200

if b > a:
  pass
print("passed")
```

```
passed
```