Python Basics II: Python Statements

A **statement** is an instruction that a Python interpreter can execute. A simple statement is comprised within a single logical line, while a compound statement, containing (groups of) other statements, generally spans multiple lines. In the following tutorial, you will learn about several Python compound statements, such as **if** statements, **for** statements, and **while** statements. You can find a list of more Python statements at [https://docs.python.org/3/reference/index.html](https://docs.python.org/3/reference/index.html).

**If statements**

The **if** statement is used for conditional execution. When you want to execute a code only if a certain condition is satisfied, decision making is required. The **if** statement is used in Python for decision making.

The syntax of the **if**, **if...else**, and **elif** statements is as follows:

```
    if expression:
        statement (s)
    elif:
        statement (s)
    else:
        statement (s)
```

The **if** keyword is used to create conditional statements and allows you to execute a block of code only if a condition is True.
The **elif** keyword is short for else if. There can be as many elif conditions as necessary between the if condition and the else conclusion.

The **else** keyword decides what to do if the condition is False.

Python supports the usual logical conditions from mathematics:

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Math Symbol</th>
<th>Python Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than</td>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>Greater than</td>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>Less than or equal to</td>
<td>≤</td>
<td>&lt;=</td>
</tr>
<tr>
<td>Greater than or equal to</td>
<td>≥</td>
<td>&gt;=</td>
</tr>
<tr>
<td>Equals</td>
<td>=</td>
<td>==</td>
</tr>
<tr>
<td>Not equals</td>
<td>≠</td>
<td>!=</td>
</tr>
</tbody>
</table>

Let’s try the following example:

```python
# check if the number is positive or negative or zero
num = int(input("Please enter an integer: "))

if num > 0:
    print("Positive number")

elif num == 0:
    print("Zero")

else:
    print("Negative number")
```

*Note: Make sure the print statements are indented.*

Press enter twice to see the output.
For Statements

The for statement is used for iterating over a sequence. Let’s say you are going grocery shopping and checking the shopping list. Lists can be created using square brackets []:

```python
shopping = ["soda", "milk", "bread"]
```

```python
for x in shopping:
    print(x, len(x))
```

The `len()` function returns the number of items in an object. When the object is a string, the `len()` function returns the number of characters in the string.

```python
len(shopping)
```

Compare `len(x)` in `shopping` and `len(shopping)` in the output:

Now you can make a script using both the if and for statements as below:

```python
students = {“Amy”: 90, “Paul”: 84, “Sally”: 59, “Dan”: 100}

for student, score in students.items():
    if score >= 90:
        print("Student : {}, Score : {}, Pass".format(student, score))
    else:
        print("Student : {}, Score : {}, Fail".format(student, score))
```
The curly braces {} are used in Python to define a dictionary. Dictionaries are used to store data values in key:value pairs. The items() method is used to return the list with all dictionary keys with values.

Another useful function is range(). Python range() function returns the sequence of the given number between the given range. Try the following:

```python
for i in range(10):
    print(i)

for i in range(1, 11):
    print(i)
```

Note that the computer counts from 0 unless you designate the starting number.
**While statements**

The while statement is used for repeated execution as long as a condition is true. Repeated execution of a set of statements is called **iteration**. If the condition is initially false, the loop body will not be executed.

The following example outputs the value of `n` until it reaches 5. The code is as follows:

```python
n = 1
while n < 6:
    print(n)
    n += 1
```

You can also make a script using both the if and while statements as bellow:

```python
while True:
    num = int(input("Please enter an integer: "))
    if num % 2 != 0:
        print("The number is odd.")
        break
    print("The number is even.")
```

The **break** statement is used to terminate the current loop and resume execution at the next statement.

The while loop starts only if the condition evaluates to True. However, if a break statement is found, the loop immediately stops. Otherwise, the loop continues its normal execution, and it stops when the condition evaluates to False.