Conditional and looping statements

Conditions and If statements

- 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.

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

Note:

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 Python's 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”)
```

Else

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

```
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")
```

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")
```

And

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

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

Or

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

Not

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

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

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.”)
```

Python Loops

Python has two primitive loop commands:

- while loops
- for loops

while Loop

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

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Note: remember to increment i, or else the loop will continue forever.

The break Statement

With the **break** statement we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

With the **continue** statement we can stop the current iteration, and continue with the next:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

For Loops

A **for** loop is used for iterating over a sequence

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Looping Through a String

```
for x in "banana":
  print(x)
```

The break Statement

With the **break** statement we can stop the loop before it has looped through all the items:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Exit the loop when **x** is "banana":

The continue Statement

With the **continue** statement we can stop the current iteration of the loop, and continue with the next:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

The range() Function

The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

```
for x in range(2, 6):
  print(x)
```

```
for x in range(2, 30, 3):
  print(x)
```

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!”)
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop”:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```