Section 2

Flow Control

Mohammed Asir Shahid

2021-07-24

Contents

1 Flow Charts and Basic Flow Control concepts

A flowchart is a series of instructions saying what to do in certain situations, generally revolving around boolean values.

1.1 Comparison Operators

`= !` $< > <= >=$

We can use these for comparisons.

```
print(42==42)
print(42=="Hello")
print(42==41)
print(2!=3)
print(42<100)
print(42>=100)
print(42<=42)
```

```
myAge=22
print("My age is {}".format(myAge))
print(myAge==22)
```

```
True
False
False
True
True
False
True
My age is 22
True
```

1.1.1 Boolean Operators

and or not

The and operator only evaluates to true if all are true.
The or operator only evaluates to true if at least one is true.
The not operator evaluates to the opposite boolean value.

```
myAge=26
myPet="cat"
print(myAge>20 and myPet=="cat")
```

```
True
```

2 If, Else, and Elif Statements

The most important portion flow control statements are the statements themselves.

```
name = "Alice"

if name == "Alice":
    print("Hi Alice")
print("Done")
```

```
Hi Alice
Done
```

Let's break this down. We have the if statement "if name="Alice"". Here, it evaluates the name variable and checks if it is true that it is equivalent to "Alice". Since it is equivalent, the program moves to the print statement.

We can look at indentation to see the differences in code blocks. Lines of code are grouped together in blocks which are dictated by indentations. We have 3 rules for code blocks: Blocks begin when the indentation increases. Blocks can contain other blocks. Blocks end when the indentation decreases to zero or to a containing block's indentation.

```
password="swordfish"

if password=="swordfish":
    print("Access granted")
else:
    print("Wrong password")
```

```
Access granted
```

Above we only print out "Access Granted" since our password variable is equivalent to "swordfish". If that was not the case, the first code block would be ignored and instead "Wrong password" would be printed out.

```
name="Bob"
age=3000
if name == "Alice":
    print("Hi Alice")
elif age<12:
    print("You are not Alice, kiddo.")
elif age>2000:
    print("Unlike you, Alice is not an undead, immortal vampire.")
elif age>100:
    print("You are not Alice, Granny")
```

```
Unlike you, Alice is not an undead, immortal vampire.
```

First the name and age variables are saved. Then the first if statement is found to be false, so it is skipped. Then the first elif block is found to be false, so it is also skipped. Then the second elif statement is found to be true, so it enters the block and prints out our statement. Then the rest of the conditions are skipped.

We can also have an else statement which will be executed if all of the above code blocks are found to be false.

```
print("Enter a name.")
#name=input()
if name:
    print("Thank you for entering a name.")
else:
    print("You did not enter a name.")
```

```
Enter a name.
You did not enter a name.
```

Why does the statement "if name" evaluate when we do not input a name? This is due to the fact that the if condition can use "truthy" or "falsey" values for strings. For example, a blank string, as seen above, would be considered falsely and would return false in the if statement. However, inputting a name would result in the if statement evaluating as true. For integers the integer 0 is the falsey value and everything else is truthy. For floats, 0.0 is the falsey value while everything else is truthy. However, it is better to be explicit and say something such as "if name != "" instead.

```
print(bool(""))
print(bool(0))
print(bool(0.0))
print(bool("Alice"))
print(bool(1))
print(bool(1.0))
```

```
False
False
False
True
```

```
True
True
```

3 While Loops

While loops can let you execute code blocks over and over again given that the while loops conditions are true.

```
spam=0
while spam<5:
    print("Hello World!")
    spam+=1
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

The above while loop iterates 5 times, until the value of spam is atleast 5.

```
spam=0
while spam<5:
    print("Hello World!")
    spam+=1
    print(spam)
```

```
Hello World!
1
Hello World!
2
Hello World!
3
Hello World!
4
Hello World!
5
```

```
spam=0
if spam<5:
    print("Hello World!")
    spam+=1
    print(spam)


Hello World!
1
```

While statements look similar to if statements, however the difference is that at the end of a if block, the program execution continues on with the rest of the program. However, with a while block at the end, it loops back to the beginning to check and see if the initial condition is still true. If so, it executes the code block again, and if it is false then it continues on with the rest of the program.

```
name = ""
while name!="your name":
    print("Please type your name.")
    name=input()
print("Thank you")
```

This would keep looping until you type out "your name".

```
spam=0

while spam<5:
    spam+=1
    if spam == 3:
        continue
    print("Spam is "  + str(spam))


Spam is 1
Spam is 2
Spam is 4
Spam is 5
```

Above we can see that when spam is 3, the string is not printed. When spam is 3, the program goes into the if block and continues.

4 For Loops

```
print("My name is")

for i in range(5):
    print("Jimmy Five Times " + str(i))
```

```
My name is
Jimmy Five Times 0
Jimmy Five Times 1
Jimmy Five Times 2
Jimmy Five Times 3
Jimmy Five Times 4
```

The code in the for loop is run 5 times. The first time it runs, i is 0, then it goes to 1, then 2, then 3, then 4.

We can write this as a while loop as well, but we would need to add some extra lines.

```
print("My name is")

i=0
while i<5:
    print("Jimmy Five Times " + str(i))
    i+=1
```

```
My name is
Jimmy Five Times 0
Jimmy Five Times 1
Jimmy Five Times 2
Jimmy Five Times 3
Jimmy Five Times 4
```

We can use for loops to do addition as follows:

```
total=0
for num in range(101):
    total+=num
print(total)
```

```
5050
```

This for loop iterates 100 times and sums up all the numbers between 0 and 101 exclusive.

```
print(range(10))
```

```
range(0, 10)
```

The range function returns a data type that contains a sequence of integers between 2 values. If only one value is given, the first number is 0. However, we can change that.

```
print("My name is")

for i in range(12,16):
    print("Jimmy Five Times " + str(i))
```

```
My name is
Jimmy Five Times 12
Jimmy Five Times 13
Jimmy Five Times 14
Jimmy Five Times 15
```

The second integer in the range function is the number that the sequence goes up to, but not included. So the above is from 12 to up to, but not including 16.

We can also add a 3rd number in order to have the sequence increase or decrease by a different amount. This is the step argument.

```
print("My name is")

for i in range(12,18,2):
    print("Jimmy Five Times " + str(i))
```

```
My name is
Jimmy Five Times 12
Jimmy Five Times 14
Jimmy Five Times 16
```

This 3rd number can be negative if we are decreasing.

```
print("My name is")

for i in range(16,12,-1):
    print("Jimmy Five Times " + str(i))
```

```
My name is
Jimmy Five Times 16
Jimmy Five Times 15
Jimmy Five Times 14
Jimmy Five Times 13
```