

Lecture 2 Notes

October 14, 2019

Lecture 2: Control Flow

In this part of the course we will be looking and boolean logic and control flow.

Boolean Variable

The Boolean data type is a data type that has one of two possible values, intended to represent the two truth values of logic and Boolean algebra. The two possible values of a boolean variable are `True` or `False`.

If Statement

An if statement is an operation which checks the condition between variables and executes code if that condition is met. Here is an example:

```
x = True
if x:
    print("Variable x is True!")
else:
    print("Variable x is False!")
```

In this example the variable `x` is checked if it has a value of `True`, if it does the first block of code will be executed. Otherwise the second block of code under `else` will be the one to be executed.

This example simply checks if the variable `x` is `True` or `False`, now we will introduce to more advanced operations

Comparison Operators

Instead of simply checking one boolean variable, using comparison operators it is possible to compare multiple variables and returning a boolean result. Here

are the operators used in Python:

```
x == y #x is equal to y
x > y #x is greater than y
x < y #x is smaller than y
x >= y #x is greater or equal than y
x <= y #x is smaller or equal than y
```

These are the most important operators, remember that each and every one of these operations returns a boolean value. For example let's take `x > y`, this operation checks if the variable `x` is greater than `y`, if `x` is greater than `y` then the operation result will be `True`, otherwise it will be `False`. Here is an example:

```
x = 10
y = 20
if x >= y:
    print("Variable x is greater or equal to y")
else:
    print("Variable x is lower than y")
```

In this case the result of the operation `x >= y` will result in `False` because `x = 10` and `y = 20`, so the block of code which will be run is the one under `else`. The output of this program will be: `Variable x is lower than y`

Elif Statement

It is possible to make multiple separate comparisons, not only `if` and `else`. This is possible using the `elif` statement. Let's look at an example using strings instead of integers:

```
name = "John"
if name == "Bob":
    print("Hi! My name is Bob")
elif name == "John":
    print("Hi! My name is John")
else:
    print("Hi! My name is neither John or Bob")
```

This block of code first checks if the variable `name` is equal to the string `"Bob"`, then it checks if it is equal to the string `"John"`, and in the end if none of previous were matched, the code under `else` will be executed. The output of this program is `Hi! My name is John`, because the operation `name == "John"` returns `True`.

Not True

If we want to check if something is not true, it is very simple. Here it is:

```

school = "IHGR"
if school != "IHGR":
    print("The school is not IHGR!")
else:
    print("The school is IHGR!")

```

The operator `!=` returns `True` when the two variables are different from each other. In this example the operation: `school != "IHGR"` will return `False` because the variable `school` is equal to the string `"IHGR"`, which means that the output of this program is gonna be `The school is IHGR!`. In Python we can also use the statement `not` to get the opposite value of a boolean result/operation.

```

x = not True
print(x)

```

This program will print `False`.

Or And

Imagine if before execturing a certain block of code you would want to check two different variables, you would have to do this by :

```

name = "Bob"
age = 19
if name == "Bob":
    if age >= 18:
        print("Barman: Here Bob, have a beer")
else:
    print("Barman: Sorry we only serve people named Bob here that are 18+")

```

In this example we are first checking if the variable `name` is equal to the string `"Bob"`, then if it is we are checking if the variable `age` is greater or equal than 18. We can put this together by using the operator `and`.

```

name = "Bob"
age = 19
if name == "Bob" and age >= 18:
    print("Barman: Here Bob, have a beer")
else:
    print("Barman: Sorry we only serve people named Bob here that are 18+")

```

Now let's say that the "barman" decides to add a new name to the list of people he serves, this is where the operator `or` will help us.

```

name = "John"
age = 19
if (name == "Bob" or name == "John") and age >= 18:
    print("Barman: Here, have a beer")

```

```

else:
    print("Barman: Sorry we only serve people named Bob or John here that are 18+")

```

Note that the `or` operation is wrapped into brackets, this because we want to first check if the `name` variable is "Bob" or "John", and if one of them is true then we are gonna check the age. The operation inside the brackets returns a boolean result which is then used into the next operation, in this case `and`.

Very Important! :

When using `and`, the operations before and after both have to be `True`.

When using `or`, only one needs to be `True` (both can be, it doesn't matter).

Exercise

Can you tell me what the output of this program is going to be?

```

member = True
age = 41
name = "Lucas"
sport = "Boxing"
if member and age >= 20:
    if (name == "Frank" or name == "Kevin") and sport != "Swimming":
        print(1)
    elif name == "John" or sport == "Boxing":
        print(2)
    else:
        print(3)
elif not member and age > 6 and age < 20:
    if name != "Bob":
        print(4)
    else:
        print(5)
else:
    print(6)

```

And what happens if we change the variables to :

```

member = False
age = 10
name = "Bob"
sport = "Swimming"

```

Challenge

- 1) Make any program that comes to your mind using as many operators as possible. Ask the user for an input, have some logic to check the input and then process that input however you want and give the user the result. Remember the course material from the previous lesson!
- 2) Create a profile maker, prompt the user for his name/age/profession etc. Make sure you check each type of input you are given, e.g. make sure the age is always an `int`. Then once you have the user data, save everything to a file. If you forgot how to work with files in python here is something to help you:

```
f = open("myfile.txt", "w")
f.write("Hello World!")
f.close()
```