Lesson 5: Control Flow

August 17, 2020

1 Lesson 5: Control Flow

In this lesson, we will learn about...

- If...Else statements
- While Loops
- For Loops

2 If...Else Statements

These allow us to selectively run parts of a program depending on if a condition is met. Let's look at the anatomy of an if statement.

```
if CONDITION:
    do something
elif CONDITION:
    do another thing
else:
    do a third thing
```

This corresponds to the following actions in plain English:

1. If the first condition is true, "do something", then skip the rest of the if...else
2. Otherwise, if the second condition is true, "do another thing", then skip the rest of the if...else
3. If none of the above conditions are true, "do a third thing".

NOTE: Only the `if` is required. You can leave out `elif` and `else` entirely, and have as many `elifs` as you want. You can only have one `if` and one `else`.

[1]:
```
x = 5
if x % 2 == 0:
    print(x, "is even!")
else:
    print(x, "is odd!")
```

```
5 is odd!
```

3 Conditions

Conditions are anything that resolve to either `True` or `False`. They can be...

- Boolean variables
- Boolean Expressions / Comparisons
- Function Calls

3.1 Falsy Values

Some values aren't the literal `False` but still evaluate to be `False`. These are called **falsy**. They are...

- Empty collections (lists, tuples, dictionaries, strings, ...)
- The number 0 in any form
- `None`
- `False`

Any other values are considered **truthy** and evaluate to `True`.

3.2 Comparisons and Logical Operators

Recall from Lesson 3 that there are several operators used for comparisons. There are also operators to combine expressions, such as `and` and `or` called logical operators.

4 Loops

Loops let us run a section of code over and over again. There are two main types of loops in python...

- While Loops
- For Loops

4.1 While Loops

These loops run **while** a condition is true. These conditions are the same as with if statements.

[2]:
```python
i = 1
while i < 6:
    print(i)
    i += 1
```

```
1
2
3
```

```
4
5
```

4.1.1 `break` and `continue`

You can use these keywords in a loop to stop the loop (`break`) or skip to the next iteration (`continue`).

[3]:
```python
i = 1
while i < 6:
    if i == 3:
        i += 1
        continue
    elif i == 5:
        break
    print(i)
    i += 1
```

```
1
2
4
```

4.2 For Loops

For loops are used to iterate over a sequence or finite range (list, tuple, set, string, ...)

[4]:
```python
fruits = ["banana", "orange", "strawberry"]

for fruit in fruits:
    print(fruit)
```

```
banana
orange
strawberry
```

4.2.1 `range()`

The `range()` function generates a sequence of numbers from 0 (by default) to a number with a step of 1 (by default).

- 1 parameter: 0 to $x - 1$
- 2 parameters: x to $y - 1$
- 3 parameters: x to $y - 1$ in increments of z

[5]:
```python
for x in range(3):
    print(x)
```

```
0
1
2
```

[6]:
```
for y in range (1,4):
    print(y)
```

```
1
2
3
```

[7]:
```
for z in range (2,10,2):
    print(z)
```

```
2
4
6
8
```

NOTE: `break` and `continue` also work here in the same way as `for` loops.