Programming with Python

Lesson 4: Lists!

November 22th, 2016

Last week's goals

Last week's goals

- We have learnt about opening and closing files

Last week's goals

- We have learnt about opening and closing files
- We have learnt to write to, and read from files using the file object

Last week's goals

- We have learnt about opening and closing files
- We have learnt to write to, and read from files using the file object
- We have learnt about the importance of functions and keeping code clean

Last week's goals

- We have learnt about opening and closing files
- We have learnt to write to, and read from files using the file object
- We have learnt about the importance of functions and keeping code clean
- We have used functions to make our code more readable, as well as to write to a log

Goodbye calculator - you won't be missed

Our calculator is finally complete!

Goodbye calculator - you won't be missed

Our calculator is finally complete!
Demonstration

A recap on variables

- An integer (1,2,3 etc)

A recap on variables

- An integer (1,2,3 etc)
- A character ('a','b','c' etc)

A recap on variables

- An integer (1,2,3 etc)
- A character ('a','b','c' etc)
- A string ("Hello, World!", "I love python xo" etc)

A recap on variables

- An integer (1,2,3 etc)
- A character ('a','b','c' etc)
- A string ("Hello, World!", "I love python xo" etc)
- A float (1.45345, 24.4562389, 7.4234 etc)

A recap on variables

- An integer (1,2,3 etc)
- A character ('a','b','c' etc)
- A string ("Hello, World!", "I love python xo" etc)
- A float (1.45345, 24.4562389, 7.4234 etc)

One more type of variable - the list

A list is a collection of variables.

One more type of variable - the list

A list is a collection of variables.
It can be something like this:

One more type of variable - the list

A list is a collection of variables.
It can be something like this:
[1, 1, 2, 3, 5, 8, 13]

One more type of variable - the list

A list is a collection of variables.
It can be something like this:
[1, 1, 2, 3, 5, 8, 13]
It can be something like this:

One more type of variable - the list

A list is a collection of variables.
It can be something like this:
[1, 1, 2, 3, 5, 8, 13]
It can be something like this:
['h', 'e', 'l', 'l', 'o']

One more type of variable - the list

A list is a collection of variables.
It can be something like this:
[1, 1, 2, 3, 5, 8, 13]
It can be something like this:
['h', 'e', 'l', 'l', 'o']
It can even be something like this:

One more type of variable - the list

A list is a collection of variables.
It can be something like this:
[1, 1, 2, 3, 5, 8, 13]
It can be something like this:
['h', 'e', 'l', 'l', 'o']
It can even be something like this:
["hello world", [1,5,7], "my name is casper", 'a', "holy macaroni", 2.542]

List operations

To get an element from a list, you use square brackets:

List operations

To get an element from a list, you use square brackets:

```
x = [1,3,5,7,9]
print(x[0])
```

List operations

To get an element from a list, you use square brackets:

```
x = [1,3,5,7,9]
print(x[0])
```

This prints the value '1', as lists are zero-indexed (the list starts at element '0' and goes up from there).

Printing out a list

We can use a while loop to print each element in a list:

Printing out a list

We can use a while loop to print each element in a list:

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

loop_counter = 0

while(loop_counter < 5):
    print("Item number " + str(loop_counter))
    print("Is a " + stuff_in_my_room[loop_counter])
    loop_counter = loop_counter + 1
```

Printing out a list

We can use a while loop to print each element in a list:

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

loop_counter = 0

while(loop_counter < 5):
    print("Item number " + str(loop_counter))
    print("Is a " + stuff_in_my_room[loop_counter])
    loop_counter = loop_counter + 1
```

However, the details are quite messy. We need to keep a loop counter variable and move it around.

Printing out a list

We can use a while loop to print each element in a list:

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

loop_counter = 0

while(loop_counter < 5):
    print("Item number " + str(loop_counter))
    print("Is a " + stuff_in_my_room[loop_counter])
    loop_counter = loop_counter + 1
```

However, the details are quite messy. We need to keep a loop counter variable and move it around.
We also have a fixed *magic* number in the example, 5.

The *len()* function

The *len()* function takes in a list of something and returns the length of the list.

The *len()* function

The *len()* function takes in a list of something and returns the length of the list.
For example, *len([1, 2, 4, 8, 16])* returns *5* as there are 5 elements in [1, 2, 4, 8, 16].

The *len()* function

The *len()* function takes in a list of something and returns the length of the list.
For example, *len([1, 2, 4, 8, 16])* returns *5* as there are 5 elements in [1, 2, 4, 8, 16].
We can use this to remove the *magic* number from our code.

No more magic numbers

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

loop_counter = 0

while(loop_counter < len(stuff_in_my_room)):
    print("Item number " + str(loop_counter))
    print("Is a " + stuff_in_my_room[loop_counter])
    loop_counter = loop_counter + 1
```

No more magic numbers

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

loop_counter = 0

while(loop_counter < len(stuff_in_my_room)):
    print("Item number " + str(loop_counter))
    print("Is a " + stuff_in_my_room[loop_counter])
    loop_counter = loop_counter + 1
```

However, we still have to fiddle around with this *loop counter*

Introducing the for loop

Along with the while loop, if we want to loop, we can use the for loop

Introducing the for loop

Along with the while loop, if we want to loop, we can use the for loop
Whereas the while loop would keep looping while a boolean was satisfied, the for loop instead does one *loop* for each value in a list

A for loop in action

This is how the previous example looks like with a for loop:

A for loop in action

This is how the previous example looks like with a for loop:

```
stuff_in_my_room = ["Bed", "Lamp", "Chair", "Computer", "Wardrobe"]

for item in stuff_in_my_room:
    print("I have a " + item)
```

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7]

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]
- [1,2,3].append(78)

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]
- [1,2,3].append(78) = [1,2,3,78]

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]
- [1,2,3].append(78) = [1,2,3,78]
- [1,2,3].pop()

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]
- [1,2,3].append(78) = [1,2,3,78]
- [1,2,3].pop() = [1,2]

Neat list functions

Below are some neat list functions you may (or may not) need when using lists:

- [1,2,3] + [4,5,7] = [1,2,3,4,5,7]
- [1,2,3] * 4 = [1,2,3,1,2,3,1,2,3,1,2,3]
- [1,2,3].append(78) = [1,2,3,78]
- [1,2,3].pop() = [1,2]

Note, if there's ever something you want to do which you are stuck on, google has tonnes of posts about pretty much everything to do with python.

With variables, loops, conditionals, file I/O and now lists we have a large and evergrowing toolkit of utensils we can use to produce high quality code.

With variables, loops, conditionals, file I/O and now lists we have a large and evergrowing toolkit of utensils we can use to produce high quality code.
It's now time to put it all into action!

Live codin'

Live codin'
Now it's your go!

https://github.com/casper-oakley/python-lessons
The skeleton I have just written can be found in the lesson4/src directory.

https://github.com/casper-oakley/python-lessons
The skeleton I have just written can be found in the lesson4/src directory.
If you're looking for some ideas for cool things to add, why not try:

- An inventory, based off a list
- A way to save and load your character from a file
- A really intricate storyline

That's all for tonight

To summarise:

That's all for tonight

To summarise:

- We learnt about lists

That's all for tonight

To summarise:

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop

That's all for tonight

To summarise:

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop
- We learnt about some list operations

That's all for tonight

To summarise:

- We learnt about lists
- We learnt about how we can iterate on lists with a for loop
- We learnt about some list operations
- We begun writing our own text based game!

For next week

Source code plus lecture slides will be available online soon after the lesson.
If you are new to HackSocNotts, please join us on *http://hacksocnotts.slack.com*.
If you have any questions, feel free to ask now or over slack.