



#CodeYork

Session 2: Functions and Control

Recap

- Last time, we looked at:
 - Primitive Data Types (Strings, Floats, Integers, Booleans)
 - Operators and Conditionals (+, -, /, *, **, ==, !=, >, <)
 - Variables and Mutability
 - If Statements
 - Lists and Indexing
 - Splicing Lists

Questions? Speak up now!



The Schedule

1. Introduction
2. **Functions and Control**
3. Recursion and Examples
4. Two Player Games



Part 1: Introduction to Functions

Intro to Functions

- The 'verbs' of a programming language
- Can define an 'action' in code and 'perform' it at any time
 - In English, can define what 'speak' means and do it after, once everyone knows what it means
 - In Python, can define what 'foo()' means and do it after, once Python knows what it means
- Sometimes our actions need to know about the world they're in
 - eg. 'eating' needs us to know who's eating and what they're eating
- Functions can have parameters, and we can pass arguments to the function:
 - eg. `print("Hello york!")` -> "Hello york!" is the argument



Functions in Python

- Functions are defined using the “def” keyword
- Functions may or may not “return” values

```
>>> def add_one(num):  
    return num + 1
```

← Function Definition

```
>>> add_one(3)  
4
```

← Function Call

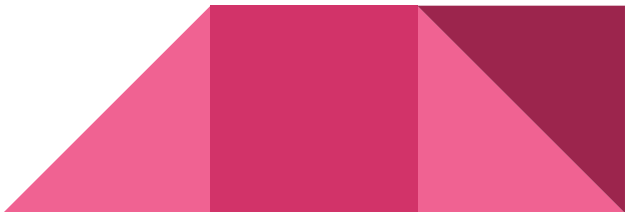
Functions Calling Functions

- Functions can call other functions if needed

```
def add_one(num):  
    return num + 1
```

```
def add_two(num):  
    return add_one(add_one(num))
```

```
print(add_two(3))
```



Python's Built-In Functions

- In the previous example, we called the “print” function
- We can call the built-in functions in exactly the same way as our own
- Some functions we've already seen are:
 - print
 - range
 - len



Course Website

<https://york.gjcampbell.co.uk/>

Task 3

```
def example(n):  
    if n > 10:  
        return True  
    elif n < 10:  
        return False
```



Task 4

```
def example(n):  
    if n > 10:  
        return square(n)  
    elif n < 10:  
        return False
```





Part 2: Further Control Structures

For Loops

- For loops to do something for every element

```
>>> for n in [1, 2, 3, 4]:  
    print(n)
```

```
1  
2  
3  
4
```



The Range Function

- The range function behaves a bit like a list

```
>>> for i in range(0, 3):  
    print(i)
```

0

1

2



While Loops

- While loops will repeat until a condition stops being true

```
>>> x = 0
>>> while x < 5:
    print(x)
    x = x + 1
```

```
0
1
2
3
4
```



The background is a solid pink color. In the top right corner, there is a decorative pattern of overlapping geometric shapes, including triangles and squares, in various shades of pink and magenta.

Go write code!

Task 3

```
>>> def foo(n):  
    while n > 1:  
        n = n / 2  
    return n
```

```
>>> foo(6)  
0.75
```



Task 4

```
>>> def sum(ls):  
    total = 0  
    for i in ls:  
        total += i  
    return total
```

```
>>> sum([1, 2, 3])  
6
```



Summary

- Today, we have looked at:
 - Defining functions
 - Calling functions
 - For loops
 - The range function
 - While loops

Questions? Speak up now!



Thanks!

Contact us:

gjc510@york.ac.uk

jr1161@york.ac.uk

<https://york.gjcampbell.co.uk/>

