



#CodeYork

Session 1: Introduction

Welcome



Graham Campbell

University of York

gjc510@york.ac.uk

<https://github.com/GrahamCampbell>



Jack Romo

University of York

jr1161@york.ac.uk

<https://github.com/jackromo>

We'll be using Python 3.6

Interpreted scripting language

Simple and readable



The Schedule

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



Part 1: Types, Operators, and Variables

Primitive Data Types 1

- Words and letters are strings (str)
 - 'egg' "spam"
- Numbers with a decimal point are floating point numbers (float)
 - 124.0 -0.123
- Numbers with no decimal point are integers (int)
 - 1 124 0 -5



Primitive Data Types 2

- Booleans give us logic
 - `True` `False`
 - Must be capitalized!
- Some things are nothing
 - `None`
 - Similar to “null” in Java.



Operators and Conditionals

- Arithmetic is nearly how you think (see exercises)

- `1 + 1 - 4 * 3 / 2 ** 2`

- Booleans can be combined

- `True and False` `True or False` `not False`

- Conditionals let us check for truth

- `1 == 1` `1 > 1`



Variables and Mutability

- Variables allow you to store values
 - `x = 5, y = "hey"`
- Python variables can hold anything
- Variables are mutable
 - `x = x + 1` (x becomes 1 greater than before)
 - `x += 1` (shorthand for above)



Course Website

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

Task 3

$$2 + 3 * 5 = 2 + (3 * 5) = 2 + 15 = 17$$

$$1 - 3 + 5 = (1 - 3) + 5 = -2 + 5 = 3$$

$$2 * 3 ** 2 = 2 * (3 ** 2) = 2 * 9 = 18$$



Task 4

`not (False and True) = not False = True`

`(not True) and True = False and True = False`

`3 and 4 = 4`

`'Foo' or 100 = 'Foo'`





Part 2: If Statements and Lists

If Statements

- These are the most common control structure you will encounter

```
if foo == 3:  
    print('Variable foo was 3!')  
else:  
    print('Variable foo was NOT 3!')
```

Note: The else clause is optional.



Lists and Indexing

- Lists can hold several items, and remember their order
- Lists are zero-indexed

```
>>> ls = [1, 2, 'hello', 3.4]
>>> ls[0]
1
>>> ls[2]
'hello'
```

Note: Lists of length n have elements 0 through $n-1$.



Splicing Lists

- Python allows you to easily take part of a list

```
>>> ls = [1, 2, 'hello', 3.4]
>>> ls[1:3]
[2, 'hello']
>>> ls[0:4]
[1, 2, 'hello', 3.4]
>>> ls[::-1]
[3.4, 'hello', 2, 1]
```



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 2

```
>>> ls = [1, 2, 'hello', 3.4]
```

```
>>> ls[0]
```

```
1
```

```
>>> ls[-1]
```

```
3.4
```



Task 3

```
>>> ls = [2, 4, 6, 8, 10]
>>> if len(ls) > 3:
    print('hi')
```

hi



Summary

- Today, we have 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!



Thanks!

Contact us:

gjc510@york.ac.uk

jr1161@york.ac.uk

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

