University of HUDDERSFIELD

CFS2160: Software Design and Development

00: The Introduction

Why learn to program?

Tony Jenkins
A.Jenkins@hud.ac.uk

Hello, World

Welcome to CFS2160.

In this module, you will learn to write programs.

```
print ('Hello, World')
```

Hello, World

Welcome to CFS2160.

In this module, you will learn to write programs.

That over there 

```
print ('Hello, World')
```

is a simple program.

Can you see what it does?

Why Learn to Program?

Programming is *absolutely fundamental* to all aspects of Computing.

Why Learn to Program?

Programming is *absolutely fundamental* to all aspects of Computing.

- ➢ You may want to become a software developer, which is all about programming.
- ➢ You may want to manage software projects, in which case you need to know about programming.
- ➢ You may want to enter another field entirely, in which case programming will almost certainly help you.

In 2018, programming ("coding") is as important a Reading, Writing, and Arithmetic.

Why Learn to Program?

Programming is *absolutely fundamental* to all aspects of Computing.

- ➢ You may want to become a software developer, which is all about programming.
- ➢ You may want to manage software projects, in which case you need to know about programming.
- ➢ You may want to enter another field entirel
  almost certainly help you.

In 2018, programming ("coding") is as important

Programming is also cool, and can even be fun.

In a Nutshell

We will start learning to program using the Python language.

- Python is widely used in industry, is a marketable skill, and is easy to get "up and running" with quickly.
- Python is also kind of cool.

Later, we will take what we know and apply it to the Java language.

- Java is (arguably) the most widely used language in industry, and therefore a highly marketable skill, but needs a bit of background knowledge.

So, we are learning to **program**; we are not learning some specific language.

How It Works

You have one lecture a week.

You also have one practical.

You *must* also allocate about four hours (an afternoon, an evening) a week to reading the book, practising, and practising some more.

How It Works

You have one lecture a week.

You also have one practical.

You *must* also allocate about four hours (an afternoon, an evening) a week to reading the book, practising, and practising some more.

If you don't, the experience of many years tells me that you will struggle.

How It Works

You have one lecture a week.

You also have one practical.

You *must* also allocate about four hours (an afternoon, an evening) a week to reading the book, practising, and practising some more.

If you don't, the experience of many years tells me that you will struggle.

Really. I mean that.

Tony

Emails:

A.Jenkins@hud.ac.uk
tony@pythonic.org.uk

Been teaching programming for, ooh, 25 years. In Pascal, then C++, then C, then Java for a bit, then Python, then Java, and now Python again.

Can program in Python (well), Java (not bad), Java Spring (can get by), C++, C. Prefer Python, but Java has its place.

Uses Linux. Dislikes Windows intensely.

Never owned a Mac. Never will.

Tony

Emails:

A.Jenkins@hud.ac.uk
tony@pythonic.org.uk

Been teaching programming for, ooh, 25 years. In Pascal, then C++, then C, then Java for a bit, then Python, then Java, and now Python again.

Can program in Python (well), Java (not bad), Java Spring (can get by), C++, C. Prefer Python, but Java has its place.

Uses Linux. Dislikes Windows intensely.

Never owned a Mac. Never will.

Tony

Emails:

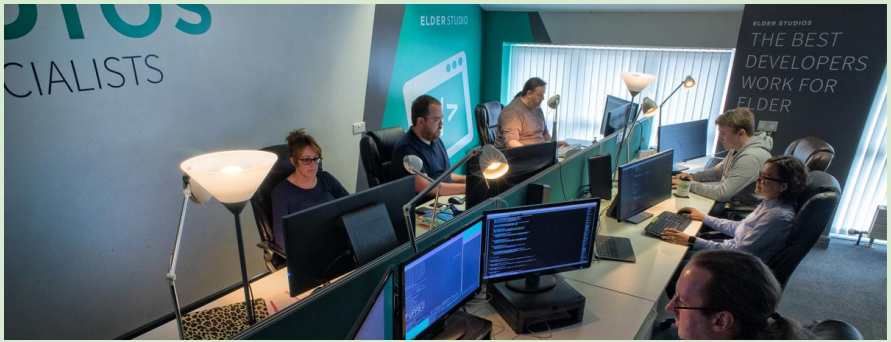
A.Jenkins@hud.ac.uk
tony@pythonic.org.uk

Been teaching programming for, ooh, 25 years. In Pascal, then C++, then C, then Java for a bit, then Python, then Java, and now Python again.

Can program in Python (well), Java (not bad), Java Spring (can get by), C++, C. Prefer Python, but Java has its place.

Uses Linux. Dislikes Windows intensely.

Never owned a Mac. Never will.



ELDER STUDIOS

Resources

You need this:

Ana Bell, "Get Programming", Manning, 2018.

Published last July.

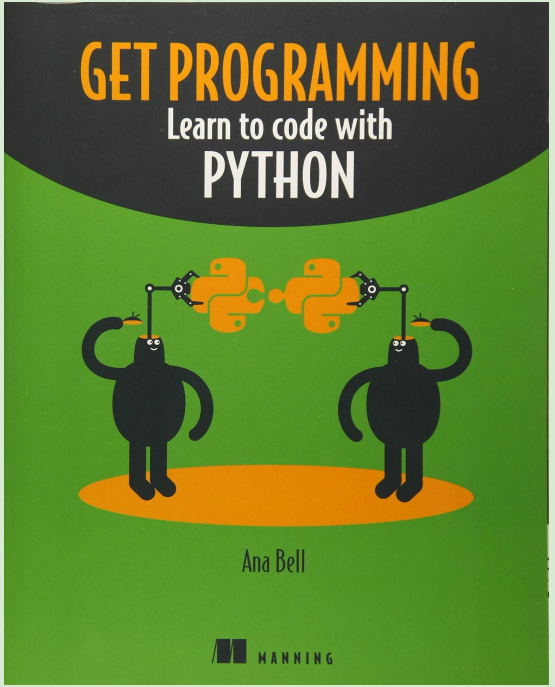
If you like eBooks, see:

https://www.manning.com/books/get-programming

If you prefer a pBook:

https://www.amazon.co.uk/Get-Programming-Learn-code-Python/dp/1617293784/

(Other online retailers exist.)



Resources

You need this:

Ana Bell, "Get Programming", Manning, 2017.

Published last July.

If you like eBooks, see:

https://www.manning.com/books/get-programming

If you prefer a pBook:

https://www.amazon.co.uk/Get-Programming-Learn-code-Python/dp/1617293784/

(Other online retailers exist.)

If you have experience in another language, this is probably not for you.
Ask your friendly tutor for a recommendation.

Resources

The official Python docs are here:

https://docs.python.org/3/

Be sure to be reading the docs for Python 3.

There are many, many online tutorials:

https://www.learnpython.org/
http://thepythonguru.com/
http://www.w3resource.com/python/python-tutorial.php

Historical Note

Historical Note

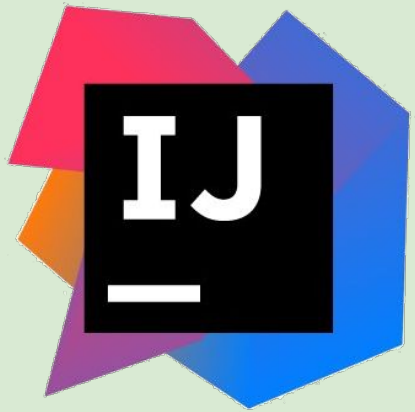
Tools

We will use PyCharm for Python programming:

https://www.jetbrains.com/pycharm/

Later, we will use IntelliJ IDEA for Java programming:

https://www.jetbrains.com/idea/



These are industry-standard tools, and therefore highly marketable skills.

They are also free for students (which is nice).

Cloud Tools

We live in the age of the Cloud.

PythonAnywhere lets you develop and run Python inside a browser.

(I may use it to show Python in lectures.)

Cloud 9 is a full IDE running in the Cloud.

(You will see me using it if I only have my ChromeBook with me.)

More Tools

Later, we will also use GitHub as a *repository* for the program code we generate.

https://www.github.com/

GitHub

Specifically:

https://education.github.com/

GitHub offer many free goodies to students.

And (guess what?) knowledge of GitHub and therefore Git is a highly marketable skill.

Resources

Git is the source code control system behind GitHub.

The official docs are here:

https://git-scm.com/

There are many, many online tutorials.

Try:

https://try.github.io/

Or the videos here:

https://git-scm.com/documentation

Or search YouTube for videos, or Google for docs.

Toolset

Any developer works best with a preferred set of tools and in a preferred environment.

This includes:

- Operating System.
- IDE / Editor.
- Keyboard / Rodent.
- Monitor (s).
- Physical Environment.
  - Quiet / Loud.
  - With coffee / Without coffee.

Your first task is to start understanding how you work best.

Toolset

Any developer works best with a preferred set of tools and in a preferred environment.

This includes:

- Operating System.
- IDE / Editor.
- Keyboard / Rodent.
- Monitor (s).
- Physical Environment.
  - Quiet / Loud.
  - With coffee~~ / Without coffee~~.

Your first task is to start understanding how you work best.

Platform Agnostic

Both Python and Java will run on any modern operating system.

- In the labs you will use Microsoft Windows.
- You will notice that I prefer to use Linux.
- You may even prefer to use Mac (many developers do).

For this module, it matters not what you use.

- PyCharm and IntelliJ are available for Windows, Linux, or Mac.
- GitHub is cloud-based.

Platform Agnostic

Both Python and Java will run on any modern operating system.

- In the labs you will use Microsoft Windows.
- You will notice that I prefer to use Linux.
- You may even prefer to use Mac (many d

For this module, it matters not what you use.

- PyCharm and IntelliJ are available for
- GitHub is cloud-based.

It's up to you.

This week is about getting to the point where we can hack some code.

Abstraction

Programming is all about *abstraction.*

A developer looks at a problem, abstracts out features, and expresses these features in a programming language.

This leads to a problem for new programmers:

- An experienced developer has seen it all before, and has much experience, so it can all look rather like magic.
- An inexperienced developer can be a bit baffled over where to start.

Abstraction

Programming is all about *abstraction.*

A developer looks at a problem, abstracts out features, and expresses these features in a programming language.

This leads to a problem for new programmers:

- An experienced developer has seen it a
  it can all look rather like magic.
- An inexperienced developer can be a b

Remember that the reason I can do this is that I've been doing it for many, many years.

And have seen it all before.

Some Concepts

Concepts: Variables

A variable is a value that we give a name.

You have probably seen the idea in maths when you did some algebra …

… or maybe in science when you looked at equations or formulas.

Concepts: Variables

A variable is a value that we give a name.

You have probably seen the idea in maths when you did some algebra …

… or maybe in science when you looked at equations or formulas.

```
miles = 150
gallons = 4.5

miles_per_gallon = miles / gallons

mark_1 = 75
mark_2 = 60

average_mark = (mark_1 + mark_2) / 2

name = 'Arthur'
title = 'King of the Britons'
```

Concepts: True and False

You might also have met this in maths.

Some statements are always True:

"Acceleration due to gravity on Earth is $9.8 m/s^2$."

Some statements are always False:

"The Moon is made of cheese."

Some statements can be True or False, depending:

"It is Thursday today."
"It is raining outside."

Concepts: True and False

You might also have met this in maths.

```
g == 9.8
```

Some statements are always True:

```
moon_made_of_cheese = False
```

"Acceleration due to gravity on Earth is $9.8 m/s^2$."

Some statements are always False:

"The Moon is made of cheese."

Some statements can be True or False, depending:

"It is Thursday today."
"It is raining outside."

Concepts: True and False

You might also have met this in maths.

Some statements are always True:

"Acceleration due to gravity on Earth is $9.8 m/s^2$."

Some statements are always False:

"The Moon is made of cheese."

Some statements can be True or False, depending:

"It is Thursday today."
"It is raining outside."

```
g == 9.8

moon_made_of_cheese = False
```

And, remember that if:
"It is raining outside" is True.
Then:
"It is not raining outside" must be False.

Concepts: Connecting Statements

A *statement* is something that is either "True" or "False":

"It is raining outside."
"I need milk."

Any two statements can be combined using words like "and" or "or":

"It is raining outside and I need milk."

The result is either "True" or "False".

Concepts: Connecting Statements

A *statement* is something that is either "True" or "False":

"It is raining outside."

"I need milk."

Any two statements can be combined using words like "and" or "or":

"It is raining outside and I need milk."

The result is either "True" or "False".

```
age = 21
has_driving_licence = True

age >= 21 and has_driving_licence

raining_outside = True
need_milk = True

raining_outside and need_milk
```

Concepts: Making Decisions

Whether a statement is True or False can form the basis of a *decision*:

"If it is raining, then I will take an umbrella."
"If I need milk, then I will go to the shop."

And statements can be combined in decisions:

"If it is not raining and I need milk, then I will go to the shop."

Concepts: Making Decisions

Whether a statement is True or False can form the basis of a *decision*:

"**If** it is raining, **then** I will take an umbrella."
"**If** I need milk, **then** I will go to the shop."

And statements can be combined in decisions:

"**If** it is not raining and I need milk, **then** I will go to the shop."

Concepts: Making Decisions

Whether a statement is True or False can form the basis of a *decision*:

"**If** it is raining, **then** I will take an umbrella."
"**If** I need milk, **then** I will go to the shop."

And statements can be combined in decisions:

"**If** it is not raining and I need milk, **then** I will go to the shop, **else** I will do without."

Concepts: Making Decisions

Whether a statement is True or False can form the basis of a *decision*:

"**If** it is raining, **then** I will take an umbrella."
"**If** I need milk, **then** I will go to the shop."

And statements can be combined in decisions:

"**If** it is not raining and I need milk, **then** I will go to the shop, **else** I will do without."

```
age = 21
has_driving_licence = True

if age >= 21 and has_driving_licence:
    can_rent_car = True
else:
    can_rent_car = False
```

Concepts: Making Decisions

Whether a statement is True or False can form the basis of a *decision*:

"**If** it is raining, **then** I will take an umbrella."
"**If** I need milk, **then** I will go to the shop."

And statements can be combined in decisions:

"**If** it is not raining and I need milk, **then** I will go to the shop, **else** I will do without."

```
age = 21
has_driving_licence = True

if age >= 21 and has_driving_licence:
    can_rent_car = True
else:
    can_rent_car = False

if not raining and need_milk:
    go_to_shop ()
else:
    do_without ()
```

Concepts: Following Instructions

There are many examples where we might follow a set of *instructions*:

- A recipe for cake.
- Directions to the station.
- Installing new software.
- Setting up a new mobile phone.

Such instructions always involve:

- Steps to follow.
- Choices or Decisions.
- Repetition.

Possibly they are expressed in a special form of language.

Concepts: Following Instructions

There are many examples where we might follow a set of *instructions*:

- A recipe for cake.
- Directions to the station.
- Installing new software.
- Setting up a new mobile phone.

Such instructions always involve:

- Steps to follow.
- Choices or Decisions.
- Repetition.

A program is just a bunch of *instructions*.

```
print ('Hello, who are you? ')
name = input ()

print ('What year were you born?')
year = int (input ())

age_this_year = 2018 - year

print ('Good to meet you, ' + name + '.')
print ('This year you will be ' + str (age_this_year) + ' years old.')

if age_this_year >= 21:
    print ('So you can rent the car.')
else:
    print ('Sorry, you are too young to rent.')
```

Concepts

So, we are going to assume that you understand:

- Variables.
- "True" and "False". (Booleans, to give them their proper name.)
- Connecting Statements with "and" or "or".
- Making Decisions.
- Following Instructions.

If you do not understand any of these, *ask* in your practical.

Communications

Communications

Email:

- A.Jenkins@hud.ac.uk

Brightspace

GitHub:

- https://github.com/TonyJenkins

LinkedIn:

- https://www.linkedin.com/in/tony-jenkins/

Communications

Email:

- A.Jenkins@hud.ac.uk

Brightspace

GitHub:

- https://github.com/TonyJenkins

LinkedIn:

- https://www.linkedin.com/in/tony-jenkins/

Important

I am part-time, so you will not find me in the building often.

Electronic means are best!

Jobs

Before next week, make sure you have:

- ➢ Got the book, and read the first chapter.
- ➢ Understood PyCharm, and run the two programs from this lecture.
- ➢ Created a student account on GitHub.
- ➢ Understood the basics of Git.

And, optionally:

- ➢ Created a student account on JetBrains, and downloaded PyCharm.
- ➢ Installed Python and PyCharm on your own system.
- ➢ Installed Git on your own system and/or hooked PyCharm up to GitHub.

Jobs

Before next week, make sure you have:

- ➢ Got the book, and read the first chapter.
- ➢ Understood PyCharm, and run the two programs from this lecture.
- ➢ Created a student account on GitHub.
- ➢ Understood the basics of Git.

And, optionally:

- ➢ Created a student account on JetBrain
- ➢ Installed Python and PyCharm on your
- ➢ Installed Git on your own system and/

There is a lot here.

If you are in doubt, or can't work something out - Ask!


python
powered