MANCHESTER
1824
The University of Manchester

PHYS20161: Introduction to Programming for Physicists

Week 1

Course leaders

- Dr Lloyd Cawthorne
  - lloyd.cawthorne@manchester.ac.uk
  - Schuster 3A.06
- Prof Clive Dickinson
  - clive.dickinson@manchester.ac.uk
  - Turing 3.126

**All of this information is available on BlackBoard.*

PHYS20161

Core course for 2nd year Physics (normal and with Astro/Theory/Europe) & 3rd year Maths/Physics students.

<u>We assume no prior experience in programming.</u>

This course was completely re-written following feedback and consultation with students. Main changes:

- Completely in Python
- Less assignments, more self-learning (with resources)

**All of this information is available on BlackBoard.*

Aims & Learning Outcomes

The aim of the course is to give a practical introduction to computer programming for physicists assuming little or no previous programming experience.

On completion successful students will:

1. Be able to write programs in Python to aid them in practical situations they will face in their degree course and future work in physics or in other fields.
2. Implement basic programming theory to write efficient code.

**All of this information is available on BlackBoard.*

Syllabus

- Elements of programming - 3 weeks
  - Introduction to Python
  - Variable types and lists
  - Operators
  - Input/output
  - Conditional expressions
  - Loops
  - Introduction to debugging, testing and errors
  - Functions
- Basic Python libraries and validation- 2 weeks
  - Python Modules
  - Introduction to math and numpy
  - Numpy arrays, built-in functions, and indexing
- Introduction to algorithms and visualisation - 3 weeks
  - Algorithms and their uses
  - Basic manipulation and visualisation of data
  - Read and write files
  - Data validation
  - Root finding
  - Basic optimisation algorithms
- Introduction to scientific programming libraries- 3 weeks
  - Advanced uses of numpy and matplotlib
  - Introduction to scipy
  - Using inbuilt functions

Anticipated changes this year

We have formalised the expected style and made the marks associated with this more transparent.

More in weeks 2 and 3.

Unanticipated changes this year

A variety of things have happened beyond our control:

- The PC labs are no longer accessible.
    - There is no way we can safely re-open these.
    - Instead we will be running online support sessions
- Independent study week has been removed.
    - This was previously used as a break for you and a chance for us to catch up on marking.
- The week after the last day of term is Christmas day
    - This might not sound important, but places severe restrictions on when we can place deadlines.
    - As a result of this, and no ISW, we have removed an assignment and changed the structure of the course.

Otherwise, this course has been largely unaffected as our assessment is remote.

Synchronous & Asynchronous materials

Like all courses this semester, we have moved to a blended learning model with synchronous and asynchronous activities each week.

Synchronous materials:

- Weekly live online interactive session
- Weekly online support (virtual labs)

Asynchronous materials:

- Weekly pre-recorded videos
- Example code
- Lecture slides
- Practice quizzes
- Style guide
- BlackBoard discussion board
- Recommended text

Our weekly live session will take place on Mondays (time TBC) and each week’s content will be released on Wednesday the week before. E.g. Week 2 content will be released on Wednesday of week 1.

TurningPoint

To facilitate the interactive sessions, you can interact with
my slides through TurningPoint. You can download the app (recommended) or access via browser: responseware.eu

**All of this information is available on BlackBoard.*

Assessments: BlackBoard tests

5 Blackboard tests worth 6% each; 30% total:

- Due end of weeks 1, 2, 3, 6, & 7
- There are practice versions available on BlackBoard
- Introduce you to a variety of examples and common bugs
- Nearly all require you to code to complete
- There is a large bank of questions; don’t expect two tests to be the same
- There are practice versions available on BlackBoard

**All of this information is available on BlackBoard.*

Assessments: Assignments

There are two assignments which you have to write from scratch:

- End of week 4, 10%
    - Calculate iterative process
- Wednesday of week 11, 60%
    - Read in & validate data
    - Apply built-in functions to fit two parameters
    - Visualise result

You will be given 3 weeks to complete the first, and 3.5 weeks to complete the final assignment. Details of marking criteria will be shared then.
You will be graded on the programme's result, approach and coding style.

Schedule 19/20

Last year, we had an intense start, followed by two demanding assignments in succession after reading week.

Key
🟥 Assignment due
🟧 Quiz due
🟪 Assignment released

Weeks	Monday	Tuesday	Wednesday	Thursday	Friday
1	23/09	lab			🟧 lab
2	🟪 30/09	lab			🟧 lab
3	7/10	lab			🟧 lab
4	14/10	lab			🟥 lab
5	21/10	lab			🟧 lab
6	28/10	ISW			
7	🟪 4/11	lab			lab
8	11/11	lab			🟧 lab
9	18/11	lab			🟥 lab
10	🟪 25/11	lab			lab
11	2/12	lab			lab
12	9/12	lab			🟥 lab

Schedule 20/21

**I have shared a google calendar on Blackboard.*

This year, we have removed the 2nd assignment and will release the final assignment earlier.

For this to work, we have to share more content in weeks 5-8. The quizzes in weeks 6 & 7 will also be more involved.

Key
🟥 Assignment due
🟧 Quiz due
🟪 Assignment released

Weeks	Monday	Tuesday	Wednesday	Thursday	Friday
1	5/10	lab			🟧 lab
2	🟪 12/10	lab			🟧 lab
3	19/10	lab			🟧 lab
4	26/10	lab			🟥 lab
5	2/11	lab			lab
6	9/11	lab			🟧 lab
7	16/11	lab			🟧 lab
8	🟪 23/11	lab			lab
9	30/11	lab			lab
10	7/12	lab			lab
11	14/12	lab	🟥 16/12		lab

All of this information is available on BlackBoard.

Late submissions

Standard University late penalties apply:

- 10% per day late until no marks remain.

If you have been granted an extension we will be notified by the Department office. These will only apply to the assignments. You must still complete the quizzes on time.
Blackboard does not have this information and will send scary e-mails automatically.

If you are unsure what has been applied to your work, please contact us.

**All of this information is available on BlackBoard.*

Plagiarism & Collusion

We take academic malpractice very seriously.

All assignments are run through a similarity check that inspects beyond variable & function names and looks at the logic and structure of the code.
It is very effective.

All students, including the original author, involved in a case are liable to penalty.

If you have been working closely with a friend and are unsure if this will affect your work please contact us.

**All of this information is available on BlackBoard.*

Labs

Labs are on Tuesdays and Fridays each week running 10:00 - 13:00
& 14:00 - 17:00.

Labs start on October 6th

Demonstrators are available in the labs to help you and give you feedback.

Details of how to access the online labs are available on BlackBoard.

The demonstrators cannot help you with assessed BB tests, or write code for your assignments.

BlackBoard Discussion board

This is the place where you should ask questions about tasks or aspects of code.

Some coding queries will be redirected to the lab as that is its purpose and is more suitable for such questions.

If you e-mail me directly asking about a question about an assignment, I will ask you to post it on the discussion board where it can be viewed by everyone.

This was very popular last year.

Feedback

There are a variety of feedback mechanisms in this course.

For quizzes, there is automated feedback released after the submission deadline.

For assignments, you will receive personal written feedback from the demonstrator that marked your work.

If anything is unclear, or you have questions, you should go to the online lab in the first instance. The demonstrators will then decide if they need further guidance from myself or Clive.

Quiz due October 9th

There is an assessed quiz due the end of week 1.

We will rehearse all content in that week & there is a practice version available.

Feel free to get in touch if you need support with this.

Recommended text

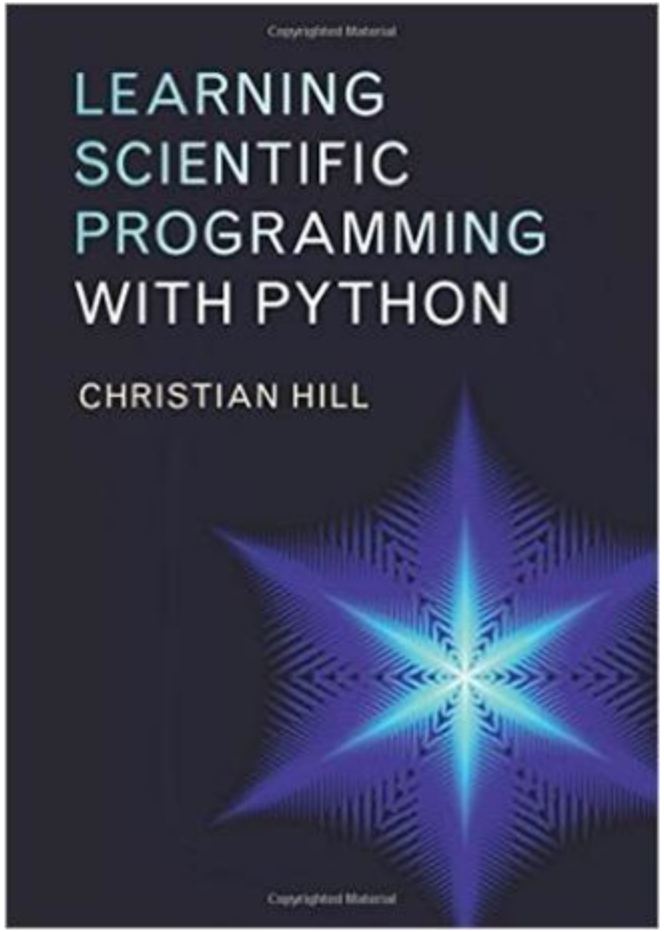
Learning Scientific Programming with Python
- Christian Hill

Contains all material needed for this course and beyond with examples.

<£30

https://scipython.com/

We can order more copies for the library if needed.



Programming languages

There are hundreds of programming languages available. We distinguish between those that need *compiling* and those that do not:

- Compiled:
    - C, C++, Java, FORTRAN, Objective-C, Lisp, etc.
- Interpreted:
    - MATLAB, IDL, Perl, Python, Ruby, PHP, Javascript, etc.

We think of all of these as different *languages* where there might be many common aspects, but could work very differently.

Once you know one language it is easier to learn more.

Python 3

We will be working in Python 3. Different to earlier versions.

It is completely opensource which makes it free & continuously evolving.

It is considered a high-level programming language in that it automatically handles fundamental operations such as memory management. It will automatically assign variables a type based on how they are used. Unlike, say C, where you have to declare what type it is in the first instance.

This makes it a good language to learn first.

However, all of these aspects running in the background can make it slow and memory intensive.

Anaconda & Spyder

There are many Integrated Development Environments (IDEs) where you can write code. The main purpose of them is to make it easier to work. They do things like autoformat, change text colour according to role, display variable values and let you debug.

We will be using Spyder which is available (for free) from the Anaconda distribution (https://www.anaconda.com/distribution/) which also includes the main libraries we will be using: numpy, matplotlib & scipy.

ANACONDA

SPYDER

Troubleshooting

Coding has a very large online community.

Many websites teach Python (e.g. https://isaaccomputerscience.org/)

Others that present neat coding challenges (e.g. https://projecteuler.net/)

And hundreds where someone wants to do something or can't get something to work and the community has presented several solutions.

Google is your friend.

Programming

..is telling the computer to do something and how to do it; it’s about giving it instructions to achieve something.

Automation – make repetitive and/or large tasks almost trivial.

Calculation – work something out quickly. Or work something out that can’t be done analytically. (numerical analysis)

Simulation – model a process (a system) and learn about it by watching how it behaves and/or tweaking parameters.

Optimisation – work out the ‘best’ way of doing something. Or the ‘best’ set of parameters to achieve a desired result.

Simple programme layout

1. IN - Machine receives from input from file, user, both or even at random.

2. DO - Machine manipulates that data. Calculation, sorting, visualising, counting, etc.

3. OUT - Machine outputs something. Number, message, graph, file, etc.

Everything happens in the order as written in the programme.

Computer Science Basics

- Computers operate with a low-level language (e.g. assembly language, object code, machine code) read in machine language (1s and 0s).
- Each element of this language is referred to as a binary digit, or bit. These can be interpreted (by us) in a variety of forms:
  - 1 or 0
  - True or False
  - Yes or No
  - On or Off
- 8 bits form one byte.

What words do you associate with ‘computer programming’?

How long does it take to download a 1Mb file at 1Mbit/s?

Numerical data storage

- We can define any base 10 number in binary (bits):
  - 0 → 0
  - 1 → 1
  - 2 → 10
  - 3 → 11
  - 4 →100
  - 5 → 101
  - Etc.

Numerical data storage

When a number is called, a computer will allocate some memory for storing these.

- int integer number composed of 32 bits (4 bytes)
- Typically ranges from -(2^{31}) to $2^{31} – 1$; or -2,147,483,648 to 2,147,483,647
- Python will update the size allocated to it automatically if needed.

Numerical data storage cont

- To work with decimals, we make use of the single-precision floating-point format, or float.
- We can define any number as *mantissa x base*^{exponent}
- e.g. 1.2345 = 1.9752000570297241 x 2^{-4}
- These occupy 32 bits of memory which are
  0   01111011 11111001101001101011011
- Sign   Exponent Mantissa
- These will be accurate to approx. 7 decimal places
- For more accuracy, we can use double which occupies 64 bits and is accurate up to approx. 16 decimals. This is the default in Python
- Speed of floating point operations is given by FLOPS, which is characteristic of the speed of the computer system.

Character and string storage

- Typically an alphanumeric character is stored as a char type of 1 byte.
- Computer then uses an encoder (ASCII, UTF-8, etc.) to relate machine language (1s & 0s) to a character (alphanumerical, greek, mathematical, etc.)

- Python instead just uses an array of chars called string.
- So a char in Python would be a string with one element.

What does the binary number 1010 correspond to?

Variable type quiz

Variable type quiz

Variable type quiz

Think like a machine quiz

A = 1

A = A*10

B = 2

C = B + A

C = C/2

D = C + B

D = ?

Think like a machine

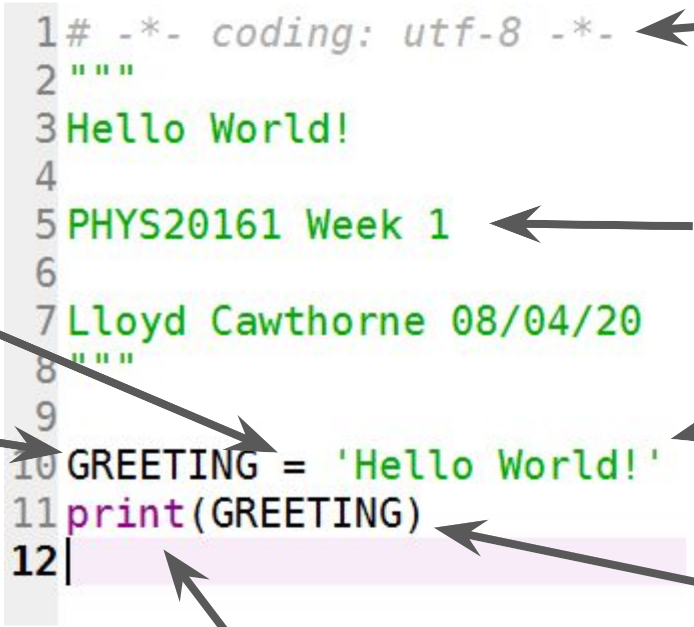
```
A = 1

A = A*10

B = 2

C = B + A

C = C/2

D = C + B
```

What is the value of D?

Variables: A, B, C, D
Labels of stored data in memory

Operators: =, *, +, /
Characters that represent an action or process, mathematical or logical

Anatomy of a programme



```python
# -*- coding: utf-8 -*-
"""
Hello World!

PHYS20161 Week 1

Lloyd Cawthorne 08/04/20
"""

GREETING = 'Hello World!'
print(GREETING)
```

Type assigning in Python

- It is the job of the language to understand the context of what is being asked. So you, as the programmer, have to ensure each type is defined correctly.
- Python assigns a type to a variable based on how it is first used.
- When the programme is run any type errors will be shown as error messages.
- We often have to *cast* between types so this runs smoothly (more later).
- If everything is fine, all that is left for the computer to do is bit manipulation.

Comments

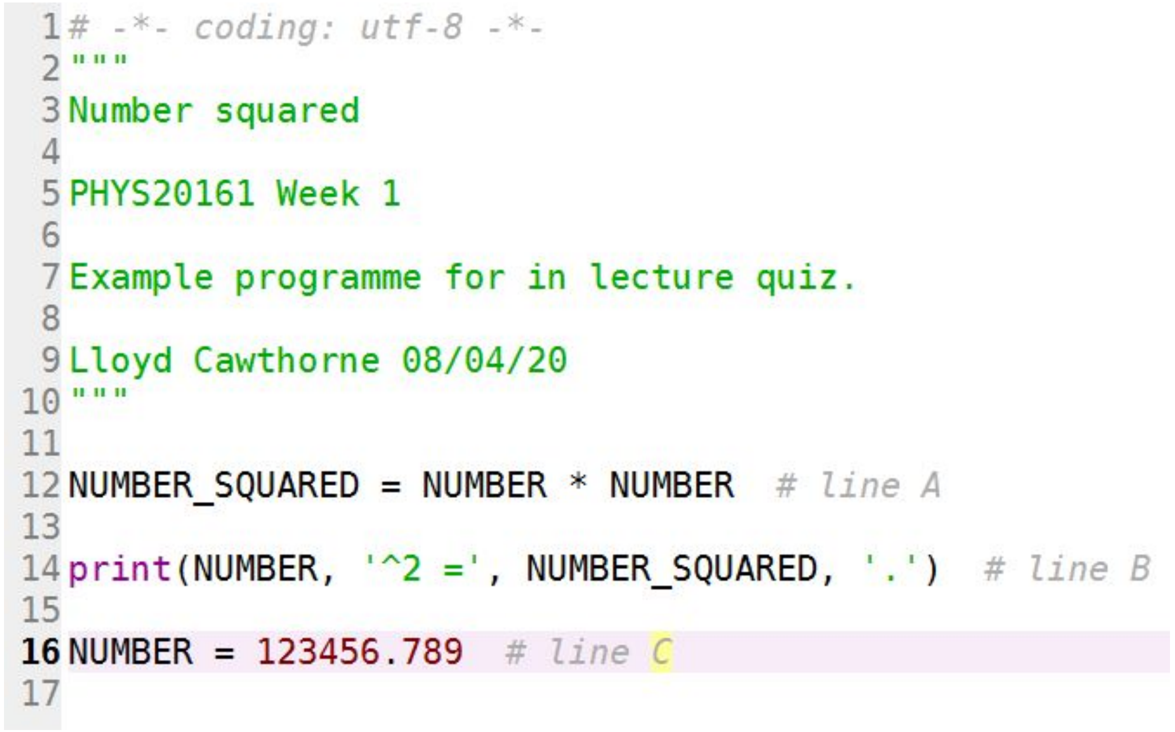
They do absolutely nothing to the programme but are paramount to understanding what is going on.

Initiate them with:

- # - Everything after on that line is a comment
- ”””...””” - Everything between, over multiple lines, is a comment

ALL programmes should start with a comment header that at least explains what the programme does, who wrote it and when. **Marks will be deducted in assignments if this is absent or incomplete.**

Programme flow quiz



```python
# -*- coding: utf-8 -*-
"""
Number squared

PHYS20161 Week 1

Example programme for in lecture quiz.

Lloyd Cawthorne 08/04/20
"""

NUMBER_SQUARED = NUMBER * NUMBER  # line A

print(NUMBER, '^2 =', NUMBER_SQUARED, '.')  # line B

NUMBER = 123456.789  # line C

```

pow()

We cannot use “ ^ “ to denote “to the power of” in Python.

Instead we can use a double asterisk, “ ** “.

Or pow().

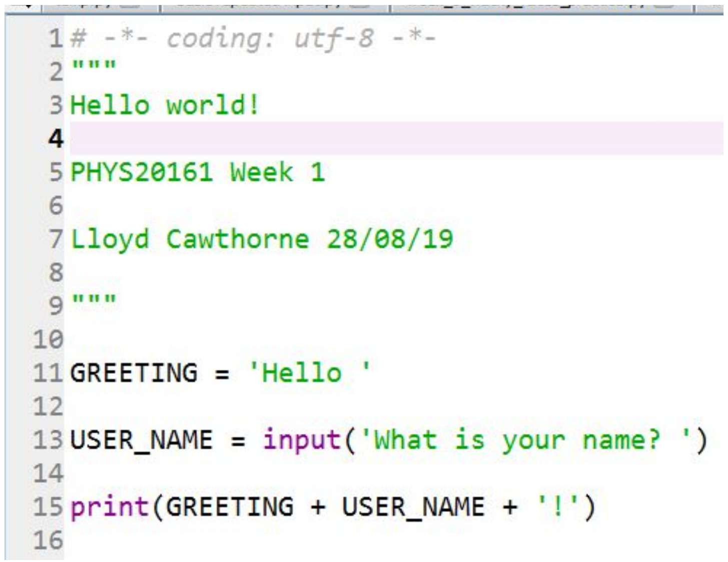
So, x^2 can be written as x**2 or pow(x, 2).

NOT x^2

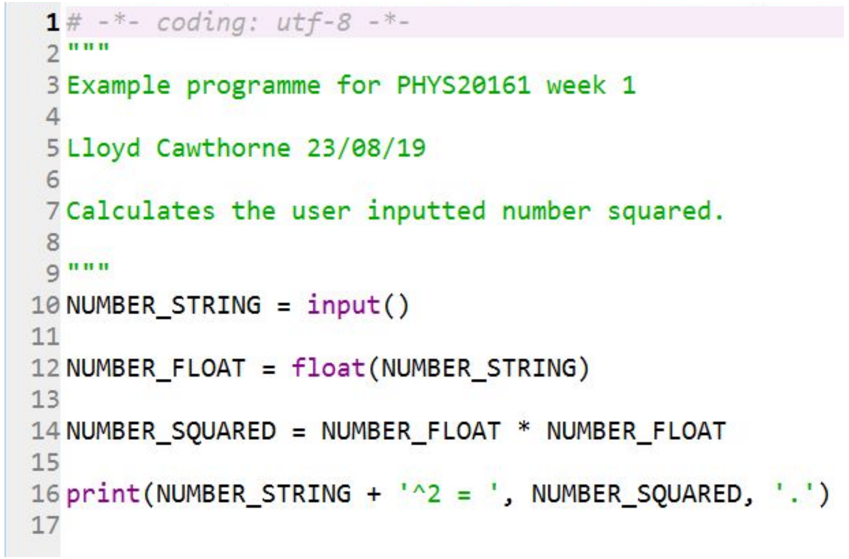
input() & print()

Basic functions to read in user data and output to screen.

Examples



```python
# -*- coding: utf-8 -*-
"""
Hello world!

PHYS20161 Week 1

Lloyd Cawthorne 28/08/19

"""

GREETING = 'Hello '

USER_NAME = input('What is your name? ')

print(GREETING + USER_NAME + '!')

```



```python
# -*- coding: utf-8 -*-
"""
Example programme for PHYS20161 week 1

Lloyd Cawthorne 23/08/19

Calculates the user inputted number squared.

"""
NUMBER_STRING = input()

NUMBER_FLOAT = float(NUMBER_STRING)

NUMBER_SQUARED = NUMBER_FLOAT * NUMBER_FLOAT

print(NUMBER_STRING + '^2 = ', NUMBER_SQUARED, '.')

```

Arrays or lists

Often we want to store lists of data.

We can also create lists of lists of data, or lists of lists of lists of data, etc.

We can call an individual element of the list by using the index.

We start counting from 0

The index used **must** be an integer.

Examples

Question

Question

Question

Summary

- Lots of information on BB on structure of this course.
- There are BB tests that are part of this course’s assessment
  - Do the practice versions first!
- Computers interpret, integers, decimals and strings differently
- We need to check the computer has interpreted the type correctly to avoid errors
- Comments allow another user to understand the code
- Arrays store multiple elements of these types.
  - We start counting at 0