CS102/CS103: Week 01 - Introductions

Dr Niall Madden
School of Mathematical and Statistical Sciences
University of Galway

This notebook is adapted from one Dr Tobias Rossmann

Welcome to CS102/CS103

This is **Introduction to Programming, which is part of CS102 and all of CS103.**

Usually I'll call it "Intro to Programming" or, with applogies to some of you, just CS102.

Introductions

Who am I?

I am Dr Niall Madden, a lecturer in Mathematics. My office is in Aras de Brun, AdB-1013.

Who are you?

This module is taken by various groups of students:

- **CS102 (1BPC1, 1BS1, 1MR1) (≈ 60 students):**

15 ECTS.
Your final mark is the average of your Semester 1 and Semester 2 scores.

- **CS103 (1BGG1, 1BMS1, 1FM1, 1PHO1) (≈ 145 students):**

5 ECTS.

The Wednesday and Thursday lectures in Semester 1 are common to both modules. These cover Introduction to Programming. For more about CS102 and CS103, see the slides from the Orientation Session posted to the "Information" section on the Blackboard modules.

But we'd like to learn more about you: please complete this survey: https://forms.office.com/r/igSzkS0rCt

Basic Information

Topics in ”Introduction to Programming”

Among other things, we will cover the following:

1. **Getting started with Python and Jupyter**
2. **Strings and text processing**
3. **Variables and expressions**
4. **Functions**
5. **Loop, iteration and recursion**
6. **Contionals and decision points**
7. **Input and output**
8. **Tuples, lists and dictionaries**
9. **Classes and objects**
10. **Files and storage**

Schedule for Intro to Programming

Lectures

Two lectures per week.

- **Wednesdays: at 12:00 in HBB-G019 and repeated at 13:00 in MRA-201.**

  Expectation is that this will change to just Wednesdays at 13:00 in MRA-201.
- **Thursday 9:00 in AMB-1022 ("Fottrell")**

Labs

Labs start in Week 3, and run every Monday, Tuesday and Wednesday 15:00-17:00 (may run to 18:00 if needed). Venues and other information next week.

Assessment (Semester 1)

- **5 programming assignments: 60%**
- **Two class/lab tests: 20% each**
- **No final exam**

No written exam this year.

Textbook

Allen B. Downey *Think Python: How to Think Like a Computer Scientist 2nd Edition*, Version 2.4.0

Freely available from https://greenteapress.com/thinkpython2/html/index.html

What is ”Programming”

What is a computer?

From Wikipedia (accessed: 28/10/2022):

> ***A computer is a digital electronic machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically. Modern computers can perform generic sets of operations known as programs. These programs enable computers to perform a wide range of tasks.***

- **A computer is a universal machine that stores and manipulates information under the control of a changeable program.**
- **A computer program is a step-by-step**

list of instructions telling a computer exactly what to do.

- **A computer is a machine for executing programs.**

Algorithms

- **In Computer Science, a *recipe* for solving a particular type of problem is called an algorithm.**
- **This course will teach you how to think algorithmically and to apply this thinking in practice.**
- **A good command of algorithms can benefit you in almost all fields of science.**

Y. N. Harari, *Homo Deus: A Brief History of Tomorrow* (2016):

> ***'Algorithm' is arguably the single most important concept in our world. If we want to understand our life and our future, we should make every effort to understand what an algorithm is [...] An algorithm is a methodical set of steps that can be used to make calculations, resolve problems and reach decisions. An algorithm isn't a particular calculation, but the method followed when making the calculation.***

Programming languages

- **Human (natural) languages are not well suited for describing complex algorithms.**
  (Can you think of some reasons?)

- **A programming language is a (mathematical) notation**

used to express computations in a precise way.

- **Every part of a program has a precise form**

(its syntax) and a precise meaning (its semantics).

- **A low-level language is designed for machine readabilty ("close to the metal").**
- **A high-level language is more human readable (e.g. more abstract or mathematical).**
- **These are not absolute terms and depend on context.**

Many high-level languages are translated ("compiled") to a low-level language prior to execution. Others are interpreted by a specialised piece of software (an interpreter).

Over the past decades, many programming languages have been conceived. Some widespread languages:

- **Assembly languages**
- **C**
- **Java**
- **Python**

All programming languages essentially serve the same purpose and have many features in common.

In this module, we use Python to learn about the fundamental elements of computer programming.

Assembly language (MIPS) example

Assembly programs consist of *extremely* low-level instructions to the processor, like

```
la $t0, a
la $t1, b
la $t2, c
lw $s0, 0($t1)
lw $s1, 0($t2)
mult $s0, $s1
mflo $s0
sw $s0, 0($t0)
```

This program computes $a = b \cdot c$.

C Example

The same formula (with $a = 12$ and $b = 127$) as a C program looks like

```c
#include <stdio.h>
  int main(void) {
    int a, b, c;
    a = 12;
    b = 127;
    c = a * b;
    printf("c = %d\n", c);
    return 0;
  }
```

Python example

In Python, the same effect can be achieved with fewer (and far more readable!) statements:

In [1]:
```
a = 12
b = 127
c = a * b
print('c =', c)
```

```
c = 1524
```

Python

Why Python?

Because Python ...

- **... is easy to learn, easy to use, and intuitive.**
- **... is powerful and in widespread use.**
- **... is a modern programming language**

(and e.g. knows about the internet).

- **... is a high-level language (like C, Java, ...).**
- **... is, by most measures, the most popular langaue in the world**

- **... is interpreted (rather than compiled).**
- **... supports object-oriented programming (but doesn't force it upon us).**
- **... is free (in every sense).**
- **... is great fun.**

See www.python.org.

History of Python

- **Guido van Rossum began to develop Python in the 1980s. He held the title of 'benevolent dictator for life' until July 2018. The name comes from *Monty Python's Flying Circus*.**
- **First public release: 1991**
- **Python 1.0: 1994**
- **Python 2.0: 2000**
- **Python 3.0: 2008. *Major changes***
- **Python 3.8: 2019**
- **Python 3.10.7: 19 September 2022**

Getting started with Python

You are not required to install Python on your own computer: there are web-based versions that you can use.

- **https://www.python.org/shell/**
- **https://repl.it/languages/python3**

Of course, you can install it (it is recommended, but not required). I suggest installing it as part of the Anaconda distribution.

Important. **There are two incompatible versions of Python in use. We'll use Python 3.**

- **For these lectures (and the assignments), we use Python from within Jupyter notebooks.**
- **Jupyter notebooks are included with the Anaconda distribution.**
- **More about them and about writing and running your own code: next week.**

Python as a calulator

We can use the Python interpreter as a powerful calculator, using

- `+`, `-`, `*`, `/` **for the usual arithmetic operations,**
- `//` **for floor division,**
- `%` **for the modulus (remainder) operation,**
- `**` **for exponentiation.**

Python can deal with very large numbers:

In [2]:
```
(12 * 4) ** (5 + 6)
```

Out[2]:
```
311640298121016115 2
```

What is the last digit of 2^{100}?

In [3]:
```
(2**100) % 10
```

Out[3]:
```
6
```

Ordinary division

In [4]:
```
1/4
```

Out[4]: 0.25

Floor division

In [5]:
```
1//3
```

Out[5]: 0

Parentheses are used for grouping as in algebraic formulae.

In [6]:
```
(3 / (2**(4+1)-5))
```

Out[6]:
```
0.1111111111111111
```

There are rules specifying the precedence and associativity of mathematical operators. (But for caution and clarity, I prefer to use parentheses).

In [7]:
```
3**(3**3)
```

Out[7]: 7625597484987

In [8]:
```
(3**3)**3
```

Out[8]: 19683

In [9]:
```
3**3**3
```

Out[9]: 7625597484987

Types of numbers (in mathematics)

There are different types of numbers. For example, the numbers we use to count objects are different from the ones we use to measure lengths.

So it is important to be aware of the different types of numbers in use, what their properties are, and what corresponds to them in Python.

Natural numbers

$0, 1, 2, 3, \ldots$**: used for counting and sorting.**

Integers

$\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$**: the natural numbers extended by negative numbers.**

Rational numbers

Positive or negative fractions consisting of a numerator and a denominator (which are both integers), like $1/2$ or $-123/456$.

Real numbers

Positive or negative numbers with a possibly infinite, possibly non-repeating decimal expansion, like $\sqrt{2}$ or π.

Complex numbers

Complex numbers are numbers of the form $a + bi$, with a *real part* a and an imaginary part b (both a and b are real numbers), where i is a square root of -1 (i.e. $i^2 = -1$).

Reminder:

$$(a + bi) + (c + di) = (a + c) + (b + d)i.$$

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i.$$

Numbers in Python

- **Python has several data types to represent and work with numerical values.**
- **These data types are disjoint:**
  each number object

has a unique data type.

- **Integers are of type `int`.**
- **In Python, integers can be arbitrarily large while still allowing for exact computations ('arbitrary-precision arithmetic').**

The type of a Python object can be determined with the `type()` function:

In [10]:
```
type(1234)
```

Out[10]:
```
int
```

In [11]:
```
123 ** 123
```

Out[11]:
```
1143743679346171900998802952280662767462180784518502297758879750523695047856668964466065683652015421696499747277306288423453431965811348959199428208744498372120994766489583590237960785490419490078072206253565269267296640648466857583828037071007667402208392 67
```

In [12]:
```
type(123 ** 123)
```

Out[12]:
```
int
```

- Rationals and reals are of type `float` ("floating point numbers").
- `float`s are only approximations to real numbers!
- If possible, use `int`s whenever precise results are required.
- Operations can be slow for huge integers.

Beware: In Python, the square of the square root of 2 is not exactly 2.

In [27]:
```
import math
math.sqrt(9)
```

Out[27]: 3.0

In [25]:
```
math.sqrt(9) * math.sqrt(4)
```

Out[25]: 6.0

- Complex numbers are of type `complex`
- Python (like engineers) uses the letter `j` for the imaginary unit

In [15]:
```
z = 2 - 3j
print(z)
```

```
(2-3j)
```

In [16]:
```
print(z.real)
print(type(z.real))
print(z.imag)
print(type(z.imag))
```

```
2.0
<class 'float'>
-3.0
<class 'float'>
```

In [30]:
```
print(type(2))
print(type(2.0))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Basic arithmetic

- **Python supports "mixed arithmetic".**
- **When a binary arithmetic operator has operands of different numeric types, the operand with the “narrower” type is widened to that of the other.**
- **Here `int` is narrower than `float`, which is narrower than `complex`.**
- **This is Python's way of realising mathematical 'inclusions', e.g. $\mathbb{Z} \subset \mathbb{Q}$ (i.e. every integer is a rational number).**

In [18]:
```
print(3 + 4)  # addition
print(3 - 4)  # subtraction
print(3 * 4)  # multiplication
print(3 / 4)  # division
print(3 ** 4) # exponentiation
```

```
7
-1
12
0.75
81
```

In [19]:
```
print(3.0 + 4)
```

```
7.0
```

In [20]:
```
float(3)
```

Out[20]:
```
3.0
```

Integer division

- **Usually, operations on two numbers of the same type result in a third number of that same type.**
- **Division of `int`s, however results in a `float`, if the corresponding fraction is not an integer.**
- **If the integer quotient is required, Python provides floor division (`//`) as a separate operation.**
- **The corresponding remainder can be computed with the modulus operator `%`.**

In [21]:
```
print(17 // 3)  # integer quotient
print(17 % 3)   # the remainder: 17 mod 3
print(divmod(17, 3)) # quotient and remainder in one
```

```
5
2
(5, 2)
```

Floor division and modulus are related: `x == (x//y)*y + (x%y)`.

In [22]:
```
print((-16) % 3)
print((-16) // 3)
```

```
2
-6
```

For further details on floor division and the modulus operation (e.g. signs), see the Python Language Reference

The `math` Library

- **Python comes with many additional modules. These extend its basic functionality.**
- **One fundamental module is the `math` library.**
- **It needs to be `import`ed before it can be used in a program.**

In [23]:
```
import math
math.sqrt(2)
```

Out[23]:
```
1.4142135623730951
```

- **The `math` library contains many useful mathematical constants and additional functions:**

| Python | Mathematics | English | |--|--|--| | `pi` | π | an approximation of the number π || `e` | e | an approximation of Euler's number e || `abs(x)` | $\lvert x \rvert$ | the absolute value of x | `sqrt(x)` | $\sqrt{x}$ | the square root of x || `sin(x)` | $\sin(x)$ | the sine of x || `cos(x)` | $\cos(x)$ | the cosine of x || `tan(x)` | $\tan(x)$ | ... || `asin(x)` | $\sin^{-1}(x)$ | the inverse sine of x || `acos(x)` | $\cos^{-1}(x)$ | ... || `atan(x)` | $\tan^{-1}(x)$ | ... || `log(x)` | $\ln(x)$ | the natural (base e) logarithm of x || `log10(x)` | $\log_{10}(x)$ | the base 10 logarithm of x || `exp(x)` | e^x | the exponential of x || `ceil(x)` | $\lceil x \rceil$ | the smallest integer $\geq x$ || `floor(x)` | $\lfloor x \rfloor$ | the largest integer $\leq x$ |

In [24]:
```
math.floor(math.pi)
```

Out[24]:
```
3
```

Summary: Numbers

- **The data type of an object determines what values it**

can have and what operations it supports.

- **Python has three built-in data types for representing numerical values: `int`, `float` and `complex`.**
- **`float`s are (only) approximations to real numbers.**
- **`int`s can be arbitrarily large.**

- **Numeric data types support the basic arithmetic operations**

addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), and exponentiation (`**`).

- **Special operations for integers are floor division (`//`)**

and remainder (`%`).

- **More functions: see the `math` library.**
- **In some situations, Python automatically converts numbers from one data type**

to another.

- **Explicit conversion (`int`, `float`, `complex`) functions can be used, too.**