M30299 – Programming
Lecture 02 – Computing with Data

Matthew Poole & Nadim Bakhshov

`moodle.port.ac.uk`

School of Computing
University of Portsmouth

2020/21

Introduction to Lecture

- The information processed by a program is known as **data**.
- In this lecture we:
  - investigate different kinds of data;
  - define what we mean by a **data type**; and
  - look at the basics of Python's **numeric** data types.
- We'll cover more on the numeric types next lecture, and look at some other data types in detail in the following few lectures.

Data

- There are many kinds of data, and different programs process data of various kinds.
- Many programs process **numerical** data.
- Commonly in programming we distinguish between:
    - **integers** (e.g. 3245, -76 and 2), and
    - **fractional numbers** (e.g. -8.3, 0.5274 and 2.0).
- There are also **strings** (e.g. `"Enter a weight in kilos"`).
- Most programs also need to use truth values (or **Boolean** values). There are only two of these: `True` and `False`.

Data types

- The terms "integer", "float" (which we use for fractional numbers), "string" and "Boolean" are **data types**.
- We often say **type**, or in some contexts **class**, instead of data type.
- Any particular data value belongs to a **single** type.
- For example, what are the types of the following values?
  - `"Hello, World!"`
  - `-57.3`
  - `12`
  - `False`
  - `9.0`
  - `"12"`

Data types

- Python's types have the names: `int`, `float`, `str` & `bool`.
- Most programming languages provide basic types such as these.
- However, there are many other kinds of data we'd like to process; for example:
    - in a graphics program: windows, textboxes, rectangles, lines, ...
    - in an adventure game: players, rooms, objects, food, ...
    - in a word processor: words, paragraphs, styles, fonts, ...
    - in a music program: tracks, albums, artists, ...
- These types are not part of any programming language; they are defined **using the language**; we'll see how later in the module.

Data types and operations

- A data type can be considered as a **set** of all its data values.
- But a data type also has some **operations** associated with it.
- For example, the numerical data types `int` and `float` both include the arithmetic operations:
    - + (addition) and − (subtraction)
    - ∗ (multiplication) and / (division)
- What other operations might be associated with `int` and `float`?
- What operations do you think the following data types have?
    - `str`
    - `bool`

`int` data values

- The integer type, `int`, comprises whole numbers (i.e. numbers without a decimal point).
- Examples of int values are:

```
312
-54
0
1424567768534345328
```

`float` data values

- The `float` data type comprises numbers with a decimal point.
- Examples of float values are:

`3.14159`	
`-27.0`	
`3.24e273`	(3.24×10^{273})
`3.24e-273`	(3.24×10^{-273})

- They are represented in **floating point** form, typically using 64 binary digits (for details, see Operating Systems & Architecture).
- The range of `float` is huge, but the accuracy is limited to about 15 significant figures.

Numeric operators - example expressions

```
>>> x = 8
>>> 2 * x + 7.5
23.5
>>> x ** 2
64
>>> x / 5
1.6
>>> x / 2
4.0
```

- Notice that the result is an `int` only when both **operands** are `int`s, but that division **always** gives a result of type `float`.

Operator precedence

- Each operator has a **precedence** which determines the evaluation order of expressions.
- Operators with higher precedence are applied before those with lower precedence.

Operator	Precedence
**	Highest (applied first)
*, /	
+, -	Lowest (applied last)

- For example:

```
>>> 1 + 2 * 3
7
```

Operator precedence

- Operators with equal precedence are applied from left to right:

```
>>> x = 10
>>> 100 / x * 2
20.0
```

- **Brackets** impose a desired order of evaluation:

```
>>> (1 + 2) * 3
9
>>> x = 10
>>> 100 / (x * 2)
5.0
```

Type conversions

- It's useful to convert values between types `str`, `int` and `float`.
- To convert a value to a particular type, we use the name of that type as a function:

```
>>> float(5)
5.0
>>> int(6.8)
6
>>> str(6.8)
'6.8'
>>> int("12")
12
>>> float("3.8")
3.8
```

Type conversions

- See that when we convert a `float` into an `int`, rounding is not performed—the part after the decimal point is simply chopped.
- We can only convert sensible values; the following will cause errors:

```
>>> float("cheese and ham")
error!
>>> int("3.8")
error! (Python doesn't convert this to 3)
```

Reading numeric values from the user

- Recall the use of `float` from the last lecture:

```
kilos = float(input("Enter a weight in kilos: "))
pounds = 2.2 * kilos
print("The weight in pounds is", pounds)
```

- The value given by the `input` function, e.g.:

```
input("Enter a weight in kilos: ")
```

  will be **a string** containing the digits inputted by the user.

- `float` then converts this string value to a float value.

Reading input from the user

- We have to use `float` in this program, since:
  - we need `kilos` to be of a numeric type;
  - the inputted value might not be a whole number.
- How would we write statements which have the following prompts?
  - What is your name?
  - How many sisters do you have?