Lecture 1 – Writing Simple Programs

M30299 Programming

School of Computing
University of Portsmouth

Introduction to Lecture

- This lecture introduces the basic steps involved in programming.
- We'll introduce a simple programming problem (a task to be performed), and then:
  - work out precisely what the task is;
  - write down a set of steps (in English) that performs the task;
  - write these steps as a Python program; and
  - check that the program works correctly.
- We'll then study the Python program in detail to introduce the basic elements of the Python language.

A programming problem

- Consider the following programming problem:

> Write a **weight converter program** that transforms a weight measured in kilos (kilograms) into an equivalent weight in pounds.

Task specification

- The first step in writing any program is to make sure that the problem is understood completely.
- For the weight conversion problem, we might simply state:
  - **User input:** a weight measured in kilos.
  - **Output to screen:** a weight measured in pounds, equivalent to the input weight (see next slide).

Specification — conversion formula

- The following equation relates kilograms and pounds:

$$pounds = 2.2 \times kilos$$

- For example,
  - 1 kilo = 2.2 pounds
  - 10 kilos = 22 pounds
- These two example conversions can be used later to **test** that the program operates correctly.

Designing the Algorithm

- The next step is to **design** an **algorithm** that accomplishes the task.
- An algorithm is a detailed sequence of actions that accomplish some task.
- Algorithms can be written in English or any other language.
- A reasonable algorithm for our task, written in English, is:
  - Obtain a kilos value from the user
  - Calculate a pounds value using $pounds = 2.2 \times kilos$
  - Output the pounds value to the screen

The Python program

- An algorithm like this can be understood/performed by a human.
- In can also be written, or **implemented**, in any programming language like C, Python or Java.
- This algorithm is implemented as a Python program as follows:

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

- We'll now test this program before studying it in detail.

Testing the program

- Let's test the program to see how it looks to the user, and to check it gives correct results:

```
Enter a weight in kilos: 1
The weight in pounds is 2.2

Enter a weight in kilos: 10
The weight in pounds is 22.0
```

Program concepts – statements

- Each line of our program is called a command or **statement**.
- The statements of a program are carried out (or **executed**) one after the other.
- Program execution ends after the last statement is executed.

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

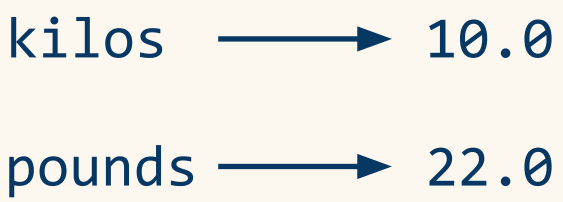
Program concepts – variables (1)

- A **variable** denotes a part of computer memory where a value is stored.
- Variables have **names** in the program; our program has two variables, `kilos` and `pounds`:

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

Program concepts – variables (2)

- We'll often draw diagrams to represent the variables and their values in the computer's memory; for example:



  says that kilos has the value `10.0` and pounds has the value `22.0`.

- A statement in the program may:
  - create a new variable;
  - access and use the value of a variable; or
  - change the value of a variable.

Program concepts – assignment statements (1)

- An **assignment statement** is used to assign a value to a variable:
  - The variable appears on the left hand side of the assignment symbol, =.
  - The right hand side is an **expression**, which has a value.
- There are two assignment statements in our program:

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

Program concepts – assignment statements (2)

- Assignment statements are executed in two steps; they:
  1. **evaluate** the expression on the right hand side (i.e. find out its value)
  2. **assign** this value to the variable on the left hand side.

- For example, for the assignment statement:

```
x = 6 + 4
```

  the following occurs:

  1. the expression `6 + 4` is evaluated to 10
  2. variable `x` is assigned the value 10.

Program concepts – assignment statements (3)

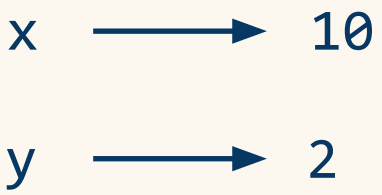
- If the variable on the left hand side doesn’t yet exist, then it is **created**.
- For example, the assignment statement:

```
y = 2
```

creates the variable y it doesn't already exist:

Before:

After:



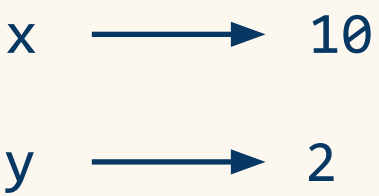
Program concepts – assignment statements (4)

- Otherwise (if the variable already exists), then its old value is replaced.
- For example, the assignment statement:

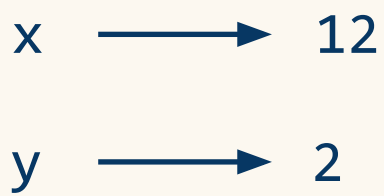
```
x = x + y
```

replaces the old value of x by a new value:

Before:



After:



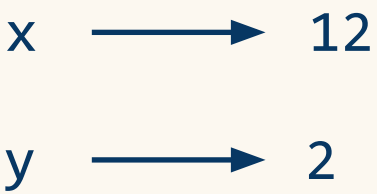
Program concepts – assignment statements (5)

- Two variables can refer to the same value (a concept known as aliasing).
- For example, the assignment statement:

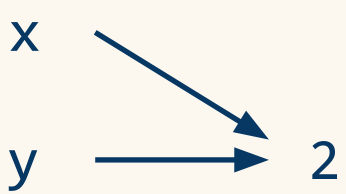
```
x = y
```

makes x refer to the same value as y.

Before:



After:



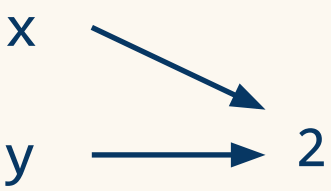
Program concepts – assignment statements (6)

- Reassigning one of the variables will remove the aliasing:

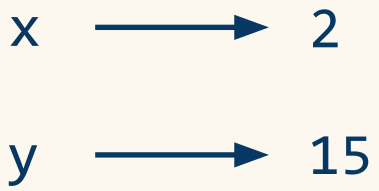
```
y = 15
```

makes y refer to a new value.

Before:



After:



Program concepts – numeric and string values

- In our program:

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

- `2.2` is an example of a **numeric** value.
- `"Enter a weight in kilos: "` is a **string** value.
- We can use double or single quotes for strings, but we can't mix them; so:
  - `"hello"` and `'hello'` are OK, whereas:
  - `"hello'` is not!

Program concepts – arithmetic expressions

- In our program we have an arithmetic expression:

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

- Python allows standard arithmetic expressions to be formed from `+`, `-`, `*` (multiplication), `/`, and brackets `(` and `)`.
- An expression is **evaluated** to give a **value**.
- We'll study arithmetic expressions in detail in the next lecture.

Program concepts – built-in functions (1)

- Our program uses three built-in functions:

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

- A **built-in function** is an algorithm that is part of the Python language, and can be accessed by using its **name**.
- The `input` built-in function:
  - Displays a prompt on the screen;
  - Waits for the user to enter a value;
  - Gives us the value that the user entered.

Program concepts – built-in functions (2)

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

- The `print` built-in function displays information to the screen.
- We'll see what the `float` built-in function does next week, where we introduce **data types** and study Python's numeric data types in detail.