M30299 – Programming

Worksheet 2: Data Types; Numeric Types

This worksheet is designed to familiarise you with Python's data types. In this worksheet, we will mainly focus on numeric data types and operations.

We assume that you understand what was covered in Worksheet 1, and have managed to complete at least the first few of its programming exercises. If this is not the case, please first spend some time studying Worksheet 1. If necessary, book a one-to-one session with the Academic Tutors (Simon Jones or Eleni Noussi) using their Moodle page. Additionally, join the Discord channel using these instructions to ask your questions.

Work through this worksheet at your own pace. You should try to complete this worksheet by next week's practical session. We will have a look at your solutions for the programming exercises in next week's practical and give you some feedback.

Python's data types

Python has a built-in `type` function which returns the data type of a given value, variable, or expression. Let's use it to find the type of some values.

Whole numbers with the `int` (integer) data type

Type the following in the shell and press enter.

Shell

```
type(3)
```

The output states that 3 is an integer (a value of type `int`).

The `type` function can also determine the type of the value stored in a variable. Test this out by creating a variable first. Enter the following in the shell:

Shell

```
my_grade = 60
```

Now check the type of the `my_grade` variable by running the following:

Shell

```
type(my_grade)
```

What about a negative number?

Shell

```
type(-64)
```

Floating-point numbers with the `float` data type

Enter the following in the shell to check the type of a number with a decimal point:

Shell

```
type(53.21)
```

Python has a `float` data type for floating-point numbers.

Does it matter if the fractional part is zero? Let's try this by creating a variable first:

Shell

```
price = 15.0
```

Now check the type of `price`. Is it still a `float` or does it become an `int`?

Text with the `str` (string) data type

Check the type of the following:

Shell

```
type("hello")
```

Textual data in Python is handled with `str` (string) data type. Strings are sequences of characters written between single quotes, 'Hello World!', or double quotes, "Bye!".

A string can include numbers:

Shell

```
up_number = '832240'
```

Now check the type of the `up_number` variable by running:

Shell

```
type(up_number)
```

True and False values with `bool` (Boolean) data type

The `bool` data type represents the truth values `False` and `True`.

Try the code below:

Shell

```
type(False)
```

This should output `bool` (short for Boolean).

Next, define a Boolean variable and check its type:

Shell

```
maths_is_fun = not True
```

Check its type by running the following:

Shell

```
type(maths_is_fun)
```

Operating on numbers

Built-in operators

The **commented values** (after the #) given below are the **expected outcomes** of the operations. Enter the code from the Example column into the shell, except for the comments, to check that they give the expected results.

Operation	Result	Example

`x + y`	sum of x and y
	$x + y$
		`x = 2.5`
		`y = 2`
		`x + y # 4.5`

`x - y`	difference of x and y
	$x - y$
		`x = 2.5`
		`y = 2`
		`x - y # 0.5`

`x * y`	product of x and y
	$x \times y$
		`x = 10`
		`y = 3`
		`x * y # 33`

`x / y`	quotient of x and y
	$x \div y$
		`x = 10`
		`y = 3`
		`x / y # 3.333...`

`x // y`	floored quotient of x and y
	(greatest integer less than or equal to $x \div y$)
		`x = 10`
		`y = 3`
		`x // y # 3`

`x % y`	remainder of x÷y
	(also known as the **modulo** operator)
		`x = 10`
		`y = 3`
		`x % y # 1`

`x ** y`	x to the power of y
	x^{y}
	
	the pow function can also be used
		`x = 3`
		`y = 2`
		`x ** y # 9`
		
		`pow(x, y) # 9`

Type conversion

We can convert between `int`s, `float`s and strings using the following built-in functions.

It is important to note that none of these functions changes the value (or type) of the variable `x` itself. They just give new values of a different type.

The table below summarises some ways we can convert between data types in Python; enter the examples into the shell to check the results.

Functions	Result	Example

`float(x)`	returns the `float` version of `x`	`x = 3`
		`float(x) # 3.0`
		
		`price_txt = '12.99'`
		`float(price_txt) # 12.99`

`int(x)`	returns the `int` version of `x` (fractional part of `float`s are removed)	`int(3.99) # 3`
		
		`message = '25'`
		`int(message) # 25`

`round(x, y)`	returns `x` rounded to the nearest `y` digits
	(without `y`, rounds `x` to the nearest `int`)
		`x = 9.87654`
		`round(x, 4) # 9.8765`
		`round(x, 1) # 9.9`
		`round(x) # 10`

`str(x)`	returns the string representation of `x`	`x = 15.00`
		`str(x) # '15.0'`
		`print(str(x)) # 15.0`
		
		`str(False) # 'False'`

Using the `math` module

Many useful functions and constants are not defined directly within the Python language. Instead, they are defined in other **modules** (Python files) that must be imported to be used in your program.

A commonly used module is the `math` module. To be able to use what this module provides, we first need to import it:

Shell

```
import math
```

Now we can use the definitions (functions and constants, for example) from the `math` module.

The table below shows some useful functions and constants from this module.

Functions	Result	Example
`math.pi`	The mathematical constant π	`math.pi # 3.14159...`
`math.sqrt(x)`	returns the square root of `x`	`math.sqrt(16) # 4.0`

`math.sin(x)`
`math.cos(x)`
`math.tan(x)`
	sine, cosine, and tangent of `x` (`x` is in radians, not degrees)	`right_angle = math.pi / 2`
		`math.sin(right_angle) # 1.0`

For more information, visit the documentation page for the `math` module.

Warm-up exercise

Let's practise using the mathematical operators that we have covered in this worksheet.

The image below shows a straight-line graph (drawn in red). We know two points on this graph: (6, 2.5), in blue, and (0, 0.5), in green.

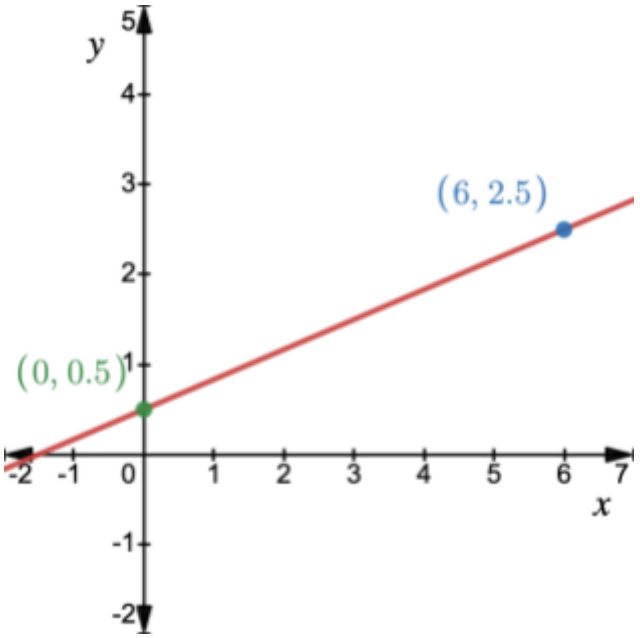

Figure 1: Two points on a straight-line graph

First type the following at the shell:

Shell

```
x_1 = 1
y_1 = 2
x_2 = 4
y_2 = 6
```

to represent two points with coordinates (1, 2) and (4, 6) in two-dimensional space (see the figure above).

Now, try to compose Python expressions that are equivalent to the following mathematical expressions. The first one gives the slope of the line that passes through the two points:

$$slope = \frac{y\_2 - y\_1}{x\_2 - x\_1}$$

The next expression uses Pythagoras' theorem to give the distance between the two points:

$$distance = \sqrt{(x\_2 - x\_1)^2 + (y\_2 - y\_1)^2}$$

If you have translated the expressions correctly, they should give values 1.333... and 5.0 respectively.

Remember to only import the `math` module once in your Python file for the programming exercises.

Additionally, you can import specific objects from a module, for example:

```
from math import sqrt, sin, cos
```

This way, you don't need to add the name of the module before the functions or constant that you use (e.g., you can use `sqrt(20)` instead of `math.sqrt(20)`).

Programming exercises

These exercises should be solved using the editor and saved to a file called `pract2.py`. We will check your answers in next week's practical.

1. Write a function `speed_calculator` that asks the user for two integers: the distance travelled in km (kilometres) and the duration of the journey in hours. `speed_calculator` should then output the average speed in kilometres per hour (km/h). The relevant formula is:

$$speed = \frac{distance}{duration}$$

2. Write a function `circumference_of_circle` that asks the user for the radius of a circle. The function should then output (print) the circle's circumference. Use the following formula. π is the `pi` constant from the `math` module, which you should import at the top of `pract2.py`.

$$circumference = 2 \times \pi \times radius$$

3. Write a function `area_of_circle` that asks the user for the radius of a circle. It should then output the circle's area.

4. Write a function `cost_of_pizza` that asks the user for the diameter of a pizza (in cm). Your function should then output the cost of the pizza (based on its area) in pounds. Assume that the cost of the ingredients is 3.5 pence per square cm.

5. Write a function `slope_of_line` that first asks the user for four values `x_1`, `y_1`, `x_2` and `y_2` that represent two points in two-dimensional space (i.e. points with coordinates (`x_1`, `y_1`) and (`x_2`, `y_2`)). The function should then output the slope of the line that connects them.

6. Write a function `distance_between_points` that asks the user for four values `x_1`, `y_1`, `x_2` and `y_2` that represent two points in two-dimensional space, and then outputs the distance between them.

7. Write a function `travel_statistics` which asks the user to input the average speed (in km/hour) and duration (in hours) of a car journey. The function should then output the overall distance travelled (in km), and the amount of fuel used (in litres) assuming a fuel efficiency of 5 km/litre.

8. Write a function `sum_of_squares` that uses a loop to output the sum $1^2 + 2^2 + \ldots + n^2$ where n is an integer provided by the user. For example, if the user enters 3, the function should output 14 ($1^2 + 2^2 + 3^2$).

   Hint: Your function should use a variable, initialised to 0 before the loop, which will hold the result when the loop finishes.

9. Write a function `average_of_numbers` which outputs the average of a series of numbers entered by the user. The function should first ask the user how many numbers there are to be inputted.

   Remember, you don't need lists or any other concept that we have not covered yet.

10. [harder] Write a function `fibonacci` which asks the user for a number n. Use a loop to calculate and output the nth value in the Fibonacci sequence.

    Hint: Once again, you don't need to use lists or if statements.

11. [harder] Write a function `select_coins` that asks the user an amount of money in pence. It should output the number of coins of each denomination (£2 to 1p) that should be used to make up that amount.

    For example, if the input is 292 pence, then the function should report the following: 1 × £2, 0 × £1, 1 × 50p, 2 × 20p, 0 × 10p, 0 × 5p, 1 × 2p, 0 × 1p

    Hint: Use integer division and the remainder (modulo). If you know lists in Python, write another version of this function, `select_coins_2`, that uses lists.