Workbook for

Introduction to Python

Exercises by **Franziska Schmidt**

Version 1.3 - December 2018

Contents

Chapter 1

Preface

The exercises in this workbook were designed for an 'Introduction to Python' course aimed at complete beginners. The course material itself is only available to course participants. If you are a course participant, you can download the lectures slides, demonstrations, and additional supporting material either from the Google Classroom or from here:
`https://www.astrofranzi.com/citylit-introduction-to-python/`.

While the exercises are best used in combination with the course, they can also be used for independent study. I made an effort to come up with exercises you won't find in books or online, so hopefully you should be able to find new projects even if you've been using other online resources.

The topics included in the workbook are:
Basic operations, variables, output and input, string operators, math operators, libraries, logical operators, IF and ELSE statements, lists, loops, and file operations.
Note that neither *functions* or *classes* are included in the syllabus. I have included a couple of extra exercises covering functions and classes at the end but they are not very extensive as they are meant to be a backup only.
While the solutions to the exercises won't make use of functions I highly recommend working with functions if you know how to.

How to use this Workbook This workbook is intended for absolute beginners with no prior knowledge of coding. If you do have some coding experience you will probably be able to skip some easier exercises. The chapter titles and hints below each exercise will point you to the knowledge required to solve the problems.
If you are not taking the course, have a look at the tutorials listed below to learn the concepts before tackling the exercises:

- `https://www.tutorialspoint.com/python/`
- `https://www.learnpython.org/`

The exercises themselves are divided into three levels of difficulty:

- **EASY**: These exercises represent a gentle introduction to a new topic. I recommend starting with these exercises unless you already feel more confident.

- **MEDIUM**: These exercises will ask you to apply what you've learned with some level of abstraction.
- **HARD**: These exercises will ask you to apply what you've learned to solve a more complex problem and are intended for learners looking for something a bit more challenging.

Additionally, some exercises will be marked with a red **M**. This indicates that the exercises contain more advanced mathematical concepts that might be unsuitable for some learners. Feel free to skip these if you're not comfortable with maths.

Solutions The workbook and solutions are available on my public GitHub account: `https://github.com/fdschmidt/PythonExercises`. All solutions are provided in the form of Jupyter Notebooks and Python scripts.
Solutions to exercises are tailored to the content covered in the course. If you learn using a different source, you might be able to solve the exercise more efficiently (e.g. with function). If an exercise requires input files you can download these from the same folder as the solutions.

How to Get in Touch For enquiries please get in touch via the contact form on my website or send me an email: `https://www.astrofranzi.com/contact/`.

Chapter 2

Output and Input

2.1 Difficulty: EASY

Exercise 2.E01

Determine the result of:

(a) $245 + 129$

(b) $501 - 333$

(c) $37 \cdot 11$

(d) $99/33$

(e) $15 + 777 \cdot 2$

(f) $30/(4 + 1)$

Hints: Use addition, subtraction, multiplication, and division operators. Remember that brackets can change the order in which mathematical expressions are evaluated.

Exercise 2.E02

Determine the primitive data type of the following expressions:

(a) `100`

(b) `0`

(c) `Python`

(d) `4.0`

(e) `6 / 2`

(f) `2017`

Hints: You can try to determine the primitive data type without using Python first. If you get stuck, remember the `type()` function.

Exercise 2.E03

Write a short program that produces the following output:

```
Hello World!
```

Hints: Use the `print()` function.

Exercise 2.E04

Write a short program that prints your name, your favourite food and a hobby. For example:

```
Hello.  My name is Cookie Monster.  I like cookies and my
        favourite hobby is not sharing my cookies.
```

Hints: Use the `print()` function.

Exercise 2.E05

Write a short program that produces the following output:

```
Hello World!
```

But this time, use a **variable**.

Hints: To generate a variable you will need the assignment operator =. Once you have stored the string inside the variable you can use the `print()` function to display it.

Exercise 2.E06

Define two variables $a = 11$ and $b = 599$. Determine:

(a) $a + b$

(b) $a - b$

(c) $a \cdot 10$

(d) $b/4$

(e) $a + 15 - b \cdot 3$

Hints: You will need to initialize two variables and then apply the addition, subtraction, multiplication and division operator to them.

Exercise 2.E07:

Decide whether the following variable declarations are valid or not:

(a) `_ingredient = Milk, flour, eggs`

(b) `london-population = 8800000`

(c) `n = 3.14159`

(d) `string = 100`

(e) `apples_amount = 10`

(f) `uk_____capital = London`

(g) `123456 = 123456`

Hint: If you are uncertain, you could always try and type the variable declaration into a Jupyter Notebook. Try and run the command and check whether Python returns an error.

Exercise 2.E08

Write a short program that asks the user for their name and then simply prints it to the screen.

Hints: You will need the `input()` and the `print()` function. You will also need to use a variable.

2.2 Difficulty: MEDIUM

Exercise 2.M01

Write a code that asks the user for their name and where they were born (city and country). Greet them with a message of the following format:

```
Hello [NAME] from [CITY] in [COUNTRY]! Nice to meet you.
```

Hints: Use the `input()` function to request user input and the `print()` function to display output on the screen. Remember that you can combine strings using the + operator (even within the `print()`) function.

Exercise 2.M02

Create a python script and using one command line only reproduce the following output:

```
Night is now falling
  So ends this day
    The road is now calling
      And I must away
```

Hint: You can use the `print()` function and spaces to reproduce above output. However, there is a quicker way of doing it: You can use the string formatters `\n` and `\t` to generate new lines and tabs.

Exercise 2.M03:

In this exercise you will generate a shopping list. You already know what you want to buy: Apples, tomatoes, cucumbers, beans, and eggs. But you will need to ask the user how many apples, tomatoes, cucumber etc. they want.
Once you know the required number of each item print a shopping list that looks something like this:

```
Shopping list:
  - Apples:  10
  - Tomatoes:  5
  - Cucumbers:  1
  - Can of beans:  1
  - Eggs:  6
```

Hints: You will need the `input()` and the `print()` function several times. Alternatively, you can use the string formatters `\t` and `\n`. Remember that the `+ operator concatenates`

strings.

Exercise 2.M04

Copy the following commands into Jupyter Notebook. Run them one by one and try to understand the error message then fix them:

(a)
```
print(Goodbye World)
```
(b)
```
# Calculating 1 + 99:
print(99 + 1 =)
```
(c)
```
# We can also print several strings within
one print command:
print(Hello World + Goodbye World)
```

Hints: Python will usually show you rather accurately where the mistake is (either by giving the line or with a little arrow). Make sure to use this!

Exercise 2.M05 [M]

Determine the result of:

(a) $(5 + 45) - (70 - 4 + 11)$

(b) $10 \cdot 4^3$

(c) $\frac{15}{0.1} + \frac{0.001}{34+22}$

(d) $\frac{4}{1/5+16} / \frac{10+3}{8 \cdot 11 - 13/27}$

Hints: Rewrite the mathematical symbols into addition, subtraction, multiplication, division operators. Remember to use brackets were appropriate.

Exercise 2.M06

Write a code that asks the user for their name and their age. Print a message greeting them and telling them in which year they will turn 100.

Hints: Use the `input()` function to request user input and the `print()` function to display output on the screen. Remember that the `input()` function always returns a string. If you want to perform a mathematical operation on an input you will need to convert it into a number first using either `int()` or `float()`.

Exercise 2.S01

Kermit, the Cookie Monster, and Big Bird have found a cookie stash. Kermit takes 10 cookies, Big Bird 15 and the Cookie Monster 45. Using variables, calculate how many cookies there are in total and display the result.

Hints: You will need a variable for the number of cookies taken by each character and another variable for the total amount of cookies. Use the `print()` function to display the result but remember type conversion of a number into a string via `str()` if necessary.

Exercise 2.S02

Before a company can send out a delivery to a costumer, the name and address of the costumer need to be determined. Write a code that asks the costumer for their name, address, postcode, city and country. Then display the address on the screen.

Hints: User the `input()` function to obtain user input and the `print()` function to display information.

2.3 Difficulty: HARD

Exercise 2.H01 [M]

Write a code that requests two numbers from the user. Store the first number in variable `var1` and the second number in `var2`. Swap the values of the variables so that `var1` contains the second number and `var2` the first

(a) using an additional variable `var3`

(b) without using an additional variable

Can you think of a reason why (a) is better than (b)?

Hints: Use the `input()` function to request user input. Remember to convert the strings returned by the `input()` function into numbers using either `int()` or `float()` if you want to perform mathematically operations on them. Try playing around with additions and subtractions to exchange the variable content without having to use an additional third variable.

Chapter 3

String Operators

3.1 Difficulty: EASY

Exercise 3.E01

Initialize a variable containing the string âĂIJTriceratopsâĂİ. Using string operators

(a) Transform the string into all upper letters

(b) Transform the string into all lower letters

(c) Count how many times the letter t occurs in the string

(d) What about the letter p?

(e) Find the index corresponding to letter a

Hints: You will need the `upper()`, `lower()`, `count()`, and `find()` operators. Remember that most string operators (but not `len()`) will be applied to strings as follows: `stringVariable.stringOperator(optional argument)`

Exercise 3.E02

Write a program that asks the user for their name and country of residence. Display the input in the following format:

```
Cookie Monster (UNITED KINGDOM)
```

Hints: Use the `input()` function to request user input and the `print()` function to display information. Use the `upper()` function to transform the country into all upper case letters. Remember: You can use the + operator to concatenate strings.

Exercise 3.E03

Look at the following code fragment:

In [1]:
```
# Request user input
firstName = input("Please enter your first name: ")
lastName = input("Please enter your last name: ")
age = input("Please enter your age: ")
```

```
Please enter your first name: Bilbo
Please enter your last name: Baggins
Please enter your age: 111
```

In [4]:
```
outputString = lastName.upper()
outputString = outputString + ", "
outputString = outputString + firstName
outputString = outputString + " ("
outputString = outputString + age
outputString = outputString + ")"
```

In [5]:
```
print(outputString)
```

Which output is produced by the last line?

(a) `Baggins, BILBO (111)`

(b) `Bilbo BAGGINS (111)`

(c) `BAGGINS, Bilbo (111)`

(d) `BAGGINS, BILBO (111)`

(e) `BILBO BAGGINS 111`

(f) `BAGGINS, Bilbo()111()`

Hints: You can copy the code into Jupyter Notebook and check your understanding of the functions is correct. Make sure you understand each individual step.

Exercise 3.E04

Write a program that accepts an arbitrary user input and then lists the number of each vowel (a, e, i, o, u) in the string. Your output should look something like this:

```
The string "Kermit" contains:
  - a: 0 times
  - e: 1 times
  - i: 0 times
  - o: 0 times
```

- u: 0 times

Hints: Use the `input()` function to request user input and the `print()` function to display information. Use the `count()` operator to count the occurrences of a specific letter in the string. Remember: You can generate tabs within string using `\t`. Depending on how you display the results you might need to convert integers into strings first using the `str()` function.

Exercise 3.E05

Initialize a variable containing the string `"Velociraptor"`. Extract the following characters by calling their respective indices:

(a) `"V"`

(b) `"l"`

(c) `"a"`

(d) `"o"`

Hints: Each character in a string is assigned an index starting at 0. To extract the character corresponding to index i from a string str use: `str[i]`.

Exercise 3.E06

Write a program that asks the user for their first and last name (use two separate commands for this). Then display the input in the following format:

```
Cookie Monster [CM]
```

Where the initials are added in brackets after the name.
Hints: Use the `input()` function to request user input and the `print()` function to display information. Remember: You can extract individual characters from a string with the following syntax: `str[i]` where i is the index corresponding to the character you want to extract.

Exercise 3.E07

Initialize a variable containing the string `"Allosaurus"`. Using character indices extract the following substrings:

(a) `"Allo"`

(b) `"osau"`

(c) `"urus"`

(d) `"losauru"`

(e) `"us"`

Hints: Remember: You can extract substrings from a string str using the following syntax: `str[i:j]` where i and j are indices. Note that `[i:j]` will extract characters from index `i` up to but excluding index `j`!

3.2 Difficulty: MEDIUM

Exercise 3.M01

Write a program that asks the user for any arbitrary input and returns the number of characters in the input string.
Bonus:
Display the number of character in the string using the following format:

```
You have entered the string [STRING] which contains [NUMBER] characters.
```

Where [STRING] is the user input and [NUMBER] the number of characters within.

Hints: Use the `input()` function to request user input and the `print()` function to display information. Use the `len()` function to count the characters in a string. For the Bonus exercise: You will need to make sure all variables are of the primitive data type string otherwise the string concatenator will run into an error. To convert the length of the string (which is a number) into a string use the type conversion function `str()` (see exercise 2.M06).

Exercise 3.M02

Using the in operator, check whether:

(a) `"b"` is part of `"London"`

(b) `"Z"` is part of `"Zimbabwe"`

(c) `"V"` is part of `"Denver"`

(d) `"t"` is part of `"Tokyo"`

(e) `" "` is part of `"New York"`

(f) Request a string and a character from the user then check whether the character is contained in the string.

Hints: Remember that Python is case sensitive. Use the `input()` function to request user input.

Exercise 3.M03

Copy the following corrupted data lines into your Jupyter Notebook:

```
"In a hole inXXXXXXXXXXX the ground there lived a hobbit.XXXX Not a
 nasty,XXXXXXXXXX dirty, wet hole, filled with theXXX ends of worms
andXXXXX an oozy smell, nor yet a dry, bare, sandy hole with nothing in
it to sit down on or to eat:  it was aXXXXXXXXXXX hobbit-hole, and that
                              means comfort."
```

Using the `replace()` function, remove all occurrences of `X` from the string and display the cleaned-up line.

Hints: Removing a character all together is the same as replacing it with an empty string.

Exercise 3.M04

Look at the following code:

```
In [2]: city = "    Wellington"
        city = city.strip()
        city = city[4:8]

        result = city[3]
```

What is stored in the variable `result`?

Hints: If you are unsure you can reproduce the command lines one by one in Jupyter Notebook.

Exercise 3.M05

Ask the user to give their full name. You will obtain input strings looking like this:

`"Cookie Monster".`

Take the strings an extract the first letter of the first name and the first letter of the last name and display the entire name in the following format:

`Cookie Monster [CM]`

Where the initials are added in brackets after the name.

Hints: You can solve this individually for each dataset by counting the indices and extracting substrings manually. However, what you want is a code that can handle all data sets of this format. So instead consider what they all have in common: Blanks as separators. Use the find() operator to determine the index of the blank. Then extract a substring from the beginning of the original string to the index of the blank. This is the first name. Extract a second substring from the index of the blank to the end of the string. This is the last name. Extract the first letters of both first and last name and combine everything in your output.

Exercise 3.M05

Start with the following data set:

```
In [16]: # Initialize the data sets
         data1 = "Harry Potter 1980"
         data2 = "Hermione Granger 1979"
         data3 = "Ron Weasley 1980"
```

Extract the name and birth year of each person, then calculate how old they are now. Display your results.

Hints: Use the `find()` function to determine the index corresponding to the first blank. A substring from the beginning to that index will contain the first name regardless of how long it is. Take the remaining string without the first name and search again for the index of the space. Another substring from the beginning to this index will contain the last name and the remainder will be the year. You can then convert this final substring into a number and calculate the age. You might also find the `strip()`, `str()` and `int()` functions useful here.

Exercise 3.S01

Write a program that takes an arbitrary string and transforms it into a title. If a string is a title all words start with a capital letter. For example:
`"harry potter and the prisoner of azkaban"` is a normal string. `"Harry Potter And The Prisoner Of Azkaban"` is a string in title format.

Hints: You could use the `find()` function, extract each word individually, transform the first letter into an upper case letter and then recombine the characters. However, this seems complicated. Instead try Google. If you search for something like "python string title transform" you will already get relevant results. Checking out some of them will tell you that the function youâĂŹre looking for exists and is called title(). If you search specifically for the python string operator `title()` you will find the following website: `https://docs.python.org/2/library/stdtypes.html`. Scroll down until you find `str.title()`. The corresponding paragraph tells you how to use the function.

Exercise 3.S02

You are given a list of student email addresses. Every email address follows the same format: studentNumber@hogwarts.ac.uk, for example: 12345678@hogwarts.ac.uk.
Write a program that accepts any email address and extracts the corresponding student number.

Hints: Consider what all email addresses have in common: They all end with @hogwarts.ac.uk and everything before that is the student number. So you want to extract a substring from the

beginning up until but excluding the @ symbol. Use the `find()` function to determine the index of the @ symbol and then slice the string.

3.3 Difficulty: HARD

Exercise 3.H01

Write a program that asks the user to give you a string and then reverses it with one command.

Hints: Try to think of how you would reverse a word on a piece of paper. You would start from the end, take down the last letter then the 2nd to last letter and so on. Translate this behaviour into a command line.

Exercise 3.H02

Write a code that asks the user how many animals of a specific type (cow, pig, horse, goat, dog, and cat) are on a farm. Then, using the `format()` function, display the information you have gathered in the following format

```
"There are 50 cows, 30 pigs, 3 horses, 3 goats, 2 dogs, and 3 cats on the
                                    farm."
```

using one command line only.

Hints: Use the `input()` function to request user input and the `print()` function to display output.

Chapter 4

Math Operators and Libraries

4.1 Difficulty: EASY

Exercise 4.E01

Determine the result of:

(a) $15 + 35.6 + 20.78 - 0.01$

(b) $0.5 + (0.8 \cdot 0.5) - 0.333$

(c) $(0.5 + 0.8) \cdot (0.5 - 0.333)$

(d) $100/(400 \cdot 4)$

(e) $100/400 \cdot 4$

(f) $2.0^5 + 100$

(g) $2.0^{(5+100)}$

*Hints: Use the addition $+$, subtraction $-$, multiplication *, division $/$ and exponent ** operators. Remember that brackets can change the order in which mathematical expressions are evaluated.*

Exercise 4.E02

Copy the following definitions into Jupyter Notebook:

```
# Initialization of variables

num1 = 0.0
num2 = 100.0
num3 = 33
num4 = -10
num5 = 0.0
num6 = 999999
```

Using comparison operators, determine whether:

(a) `num1` is smaller than `num3`

(b) `num2` is smaller than or equal to `num6`

(c) `num3` is larger than `num4`

(d) `num5` is larger than or equal to `num6`

(e) `num5` and `num1` are the same

(f) `num2` and `num4` are not the same

Hints: Use the comparison operators $<$, $<=$, $>$, $>=$, $==$, *and* $!=$. *Remember that* $=$ *is a declaration and* $==$ *a comparison!*

Exercise 4.E03

In this exercise you will be using functions defined in the math library. To use them in your own code, you will first need to import the library with the following syntax:

```
import math
```

Copy the following variable declarations into a Jupyter Notebook

```
# Initialization of variables

num1 = 100
num2 = 0.0
num3 = -100
num4 = 33
num5 = 16
num6 = 1000
```

then

(a) Calculate the square root of `num1`

(b) Calculate the square root of `num4`

(c) Calculate the absolute value of `num5`

(d) Calculate the absolute value of `num3`

(e) Determine whether the absolute values of `num1` and `num3` are the same

(f) Determine whether the square root of `num6` is larger than `num1`

(g) Determine whether the absolute value of `num5` is smaller than `num2`

(h) Determine whether the square root of the sum of `num1` and `num2` is larger than 200

(i) Determine the multiplication product of the square root of `num5` and the absolutevalue of `num3`

(j) Determine the square root of the square root of `num5`

Hints: To use a function from a library you follow the syntax

```
libraryName.functionName(arguments).
```

In this exercise you will need the math.sqrt(x) function to calculate the square root of x *and the* `math.fabs(x)` *function to calculate the absolute value of* x*. Remember: You can compare numbers using the comparison operators which will return a Boolean to you. You can also carry out operation in between the brackets of a function.*

Exercise 4.E04 [M]

Translate the mathematical notations used below into familiar Python command structures and solve the equations

(a) For $x = 5$ determine the result of $y = 6x^2 + 3x + 2$

(b) For $x = 0.5$ determine the result of $y = \sqrt{\sqrt{\sqrt{x/3.6}}}$

(c) For $a = 3$, $b = 10$, and $c = -2$ determine the two possible results of $y_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Hints: In this exercise you will need additions $+$*, subtractions* $-$*, multiplications *, exponents **, square roots* `sqrt()` *(from the math library) and divisions* $/$*.*

4.2 Difficulty: MEDIUM

Exercise 4.M01

Write a code that accepts a distance and converts it from:

(a) Kilometres to miles

(b) Miles to Kilometres.

The conversion rules are given by the following equations:
Conversion km to miles:

$$x_{\text{miles}} = 0.621371 \cdot x_{\text{km}}$$

Conversion miles to km:

$$x_{\text{km}} = 1.60934 \cdot x_{\text{miles}}.$$

*Hints: You will need the multiplication * operator. To check whether your code is working, you can test it using the following website:* `https://www.rapidtables.com/convert/length/km-tomile.html`

Exercise 4.M02

Write a code that accepts a temperature and converts it from:

(a) Fahrenheit to Celsius

(b) Celsius to Fahrenheit.

The conversion rules are given by the following equations:
Conversion Fahrenheit to Celsius::

$$T_{\text{Celsius}} = (T_{\text{Fahrenheit}} - 32) \cdot 5/9$$

Conversion Celsius to Fahrenheit::

$$T_{\text{Fahrenheit}} = T_{\text{Celsius}} \cdot 9/5 + 32.$$

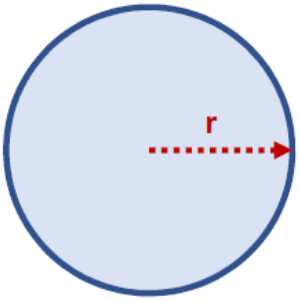
*Hints: You will need the addition $+$, subtraction $-$, multiplication * and division / operators. To check whether your code is working, visit the following website:* `https://www.rapidtables.com/convert/temperature/celsius-to-fahrenheit.html`

Exercise 4.M03 [M]

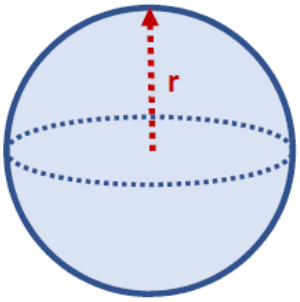
Using the `math.pi` constant from the `math` library, write a code that calculates

(a) The area of a circle given a radius r

(b) The volume of a sphere given a radius r.



The area A of a circle is calculated via $A = \pi \cdot r^2$.



The volume V of a sphere is calculated via $V = \frac{4}{3} \cdot \pi \cdot r^3$.

Hints: In addition to the `math.pi` constant, you will need the division operator `/`, the multiplication operator `*` and the exponent operator `**`. You can check your code with the following links: `https://www.calculateme.com/cArea/AreaOfCircle.htm` (area of a circle) and `https://www.calculateme.com/cVolume/VolumeOfSphere.htm` (volume of a sphere)

Exercise 4.M04

Using compound operators, determine the results of

(a) `num1 = num1 + 50`

(b) `num2 = num2 - 100`

(c) `num3 = num3 * 1000`

(d) `num4 = num4 / 10`

(e) `num4 = num4 + num1`

Hints: Before copying anything into your Jupyter Notebook, try rewriting the commands given in (a) to (e) using the compound operators. Then make sure the results check out.

Exercise 4.M05 [M]

Write a code that asks the user to provide a time duration in seconds. Convert the time into the following format: w days, x hours, y minutes and z seconds and display the result.

Hints: There are multiple ways of solving this exercise. The easiest way is via the modulus and floor division operators from the standard Python library.
Using // and % you can now approach the exercises like this: Say T is the time in seconds given by the user.

1. *The first thing you want to do is figure our how many days are in T seconds. To do that calculate:*

   ```
   days = T // (24 * 60 * 60 )
   ```

   *24 * 60 * 60 are the number of seconds in one day. By calculating the floor division T // (24 * 60 * 60) you can determine how many full days fit into T. Then you will need to figure out how many seconds are now left once you have removed the seconds that correspond to full days. To do that calculate T = T % (24 * 60 * 60) The modulus will automatically give you the remainder of the T // (24 * 60 * 60) division which corresponds to the leftover seconds.*

2. *The next step will determine how many hours are in the remaining T seconds. To do that calculate:*

   ```
   hours = T // (60 * 60)
   ```

   *60 * 60 are the number of seconds in one hour. By calculating the floor division T // (60 * 60) you can determine how many full hours fit into T. Then you will need to figure out how many seconds are now left once you have removed the seconds that correspond to full hours. To do that calculate*

   ```
   T = T % (60 * 60)
   ```

   *The modulus will automatically give you the remainder of the T // (60 * 60) division which corresponds to the leftover seconds.*

3. *Now determine the number of full minutes in the remaining seconds following the example above. One minute contains 60 seconds.*

4. *Finally, the leftover number are the remaining seconds.*

Visit: `https://www.tools4noobs.com/online_tools/seconds_to_hh_mm_ss/` to check whether your code is returning the right results.

Exercise 4.S01

Import the math library and calculate:

(a) $\sin x$ for $x = 0$

(b) $\sin x$ for $x = \frac{1}{2}\pi$

(c) $\cos x$ for $x = 0$

(d) $\cos x$ for $x = \frac{1}{2}\pi$

Hints: You can calculate them in Python using the `math.sin()` and `math.cos()` functions from the `math` library. The mathematical constant π is stored in the variable math.pi from the `math` library.
Have a look at the documentation for the math library here: `https://docs.python.org/2/library/math.html` for descriptions of the `sin()` and `cos()` functions and how to use them.

Exercise 4.S02

Write a code that converts an angle from

(a) radians to degrees

(b) degrees to radians

The conversion rules are as follows:
Conversion Radians to Degrees:

$$x_{\text{degrees}} = \frac{360}{2\pi} \cdot x_{\text{radians}}$$

Conversion Degrees to Radians:

$$x_{\text{radians}} = \frac{2\pi}{360} \cdot x_{\text{degrees}}$$

Hints: You can look up the value of π and declare it as a variable in your code or import it from the `math` library using `math.pi`. Have a look at the documentation for the `math` library here: `https://docs.python.org/2/library/math.html` for more information on the `math.pi` constant and other function included in the `math` library. To check the results, visit: `https://www.rapidtables.com/convert/number/radians-to-degrees.html` (Radians to degrees), `https://www.rapidtables.com/convert/number/degrees-to-radians.html` (Degrees to radians).

4.3 Difficulty: HARD

Exercise 4.H01

Look at the following code block:

```
In [6]: # Define variables
        num1 = 0.3
        num2 = 0.2 + 0.1

        # Check if both variables contain the same value
        print(num1==num2)

        False
```

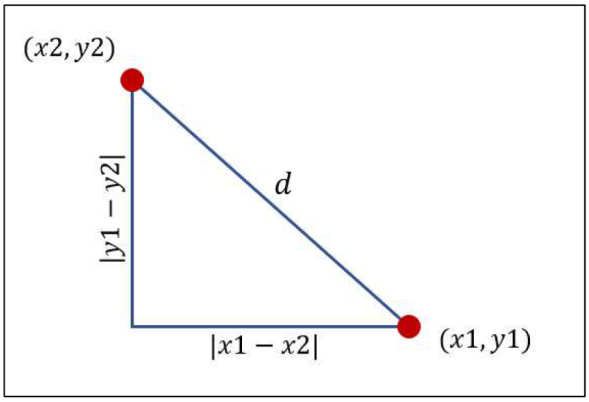
Why is the result `FALSE`? Where is the mistake coming from? How could you compare numbers without encountering this problem?

Hints: Display num1 and num2 to your screen and compare the output.

Exercise 4.H02 [M]

Write a code that measures the distance between two points in two-dimensional space. The coordinates of the first point are `x1` and `y1`, the coordinates of the second point are `x2` and `y2`. To calculate the distance d use the following equation:

$$d^2 = |x1 - x2|^2 + |y1 - y2|^2$$



Bonus:
Instead of hard coding the coordinates into your code ask the user to provide them in the following format: `(x, y)`. Extract the coordinate information from the user input using string operators and calculate the distance.

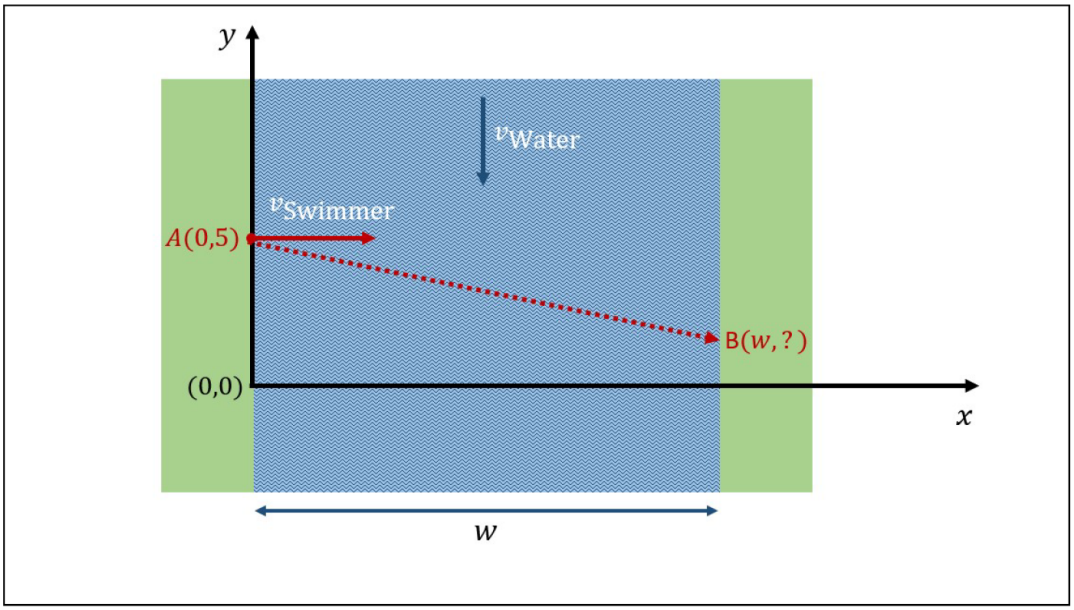
Hints: You will need the `math.fabs()` and the `math.sqrt()` functions from the `math library`.
For the bonus exercise:
Try and identify a pattern in the input format and then find a way to extract information from the string. Remember you can use the find() function to search for characters in a string. `str[i:j]` will generate a substring from string str ranging from index `i` up to but excluding index `j`. To convert a string into a number use the `str()` function.

Exercise 4.H03 [M]

Have a look at the figure below A swimmer at position A wants to cross the river of width w. The swimmer can swim with a constant velocity v_{Swimmer} but only along the x-axis. The water itself flows with velocity v_{Water} along the y-axis. Once the swimmer enters the water at position $A(0, 5)$ and starts to cross the river, the water will carry them downstream with respect to A. Write a code that calculates the position B at which the swimmer reaches the other shore for:

$$v_{\text{Swimmer}} = 0.5\,m/s,\ v_{\text{Water}} = 1.0\,m/s, \text{ and } w = 10\,m.$$

Display your result.



*Hints: In this exercise you will need the multiplication * and division / operators. The velocity v is connected to the distance d and the time t via the following equation: $v = d/t$. Try solving for d or t to obtain equations to calculate time and distance given a specific velocity v.*

Chapter 5

Logical Operators

5.1 Difficulty: EASY

Exercise 5.E01

Complete the following so-called truth table.

A	B	A AND B	A OR B	NOT A	NOT B
FALSE	FALSE				
FALSE	TRUE				
TRUE	FALSE				
TRUE	TRUE				

Hints: You can use this table throughout the remainder of the course whenever you need to generate conditional cases using logical operators!

Exercise 5.E02

Consider the variables:

```
# Initialization of variables
num1 = 10
num2 = 100
num3 = -33
num4 = 0
num5 = 50
```

and evaluate the following expressions:

(a) `(num1 > num2) or (num3 > num4)`

(b) `(num1 != num3) and (num4 < num5)`

(c) `((num4 + num5) > 0) or (num2 < num3)`

(d) `not (num2 == num4)`

Hints: You can solve this exercise on the exercise sheet. If youâĂŹre struggling to get started, copy the command lines into a Jupyter Notebook and execute the code. You can also try copying the commands onto a piece of paper and colouring corresponding brackets in the same colour.

Exercise 5.E03

Simplify the following logical expressions for an arbitrary Boolean bol:

(a) `bol and False`

(b) `bol or False`

(c) `bol and True`

(d) `bol or True`

(e) `not (not bol)`

Hints: You can solve this exercise on the exercise sheet. If youâĂŹre struggling to get started, copy the command lines into a Jupyter Notebook and execute the code for some examples of `bol`*.*

5.2 Difficulty: MEDIUM

Exercise 5.M01

Consider the following piece of code:

```
In [11]: x = 5
         y = "This is a string"

         # Case 1:
         print("x > 0 or y > 0: " + str(x > 0 or y > 0))

         x > 0 or y > 0: True

In [14]: # Case 2:
         print("x > 0 and y > 0: " + str(x > 0 and y > 0))

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-14-985f8b2081d9> in <module>()
         ----> 1 print("x > 0 and y > 0: " + str(x > 0 and y > 0))

         TypeError: '>' not supported between instances of 'str' and 'int'
```

The operation `y > 0` is invalid because `y` is a string and `>` a mathematical operator. Why is Case 1 executed without a problem while Case 2 results in an error message?

Hints: Consider the different behaviour of the `AND` and the `OR` operator when they encounter one `TRUE` statement.

Exercise 5.M02

Starting with the following variable definitions:

```
1 # Initialize the variables
2 bol1 = False
3 bol2 = True
4 bol3 = False
```

Determine the value of:

(a) `bol4 = bol1 or bol2 and not bol3`

(b) `bol5 = not bol1 and bol3 or bol2`

(c) `bol6 = not(bol2 or bol3)`

(d) `bol7 = not(bol2 and not bol3)`

(e) `bol8 = not(bol1 or bol2) or (bol2 and bol3)`

(f) `bol9 = bol1 or (bol1 or (bol1 or bol2)) or bol3`

(g) `bol10 = bol9 and bol2 and not bol3`

Hints: You can solve this exercise on the exercise sheet. If youâĂŹre struggling, try copying the relevant lines into Jupyter Notebook and execute them step by step.

Exercise 5.M03 [M]

Starting with the following variable definitions:

```
# Initialize the variables
a = 0
b = 10
c = -5
d = 3
e = 3
```

Determine the expressions::

(a) `bol1 = a > b or a > c`

(b) `bol2 = a == 0.0 or d == 4.9`

(c) `bol3 = c < -6 or (d > 0 and e == d)`

(d) `bol4 = a + b <= 11 or d == 5`

(e) `bol5 = not (a > 30 or d < 0)`

(f) `bol6 = not (not (not (a + e > 2 or c + b > 0) or e  a > 2))`

(g) `bol7 = not (b%2 == 0)`

(h) `bol8 = (c%2 > 0 or e%2 > 0 or b%2 > 0) and a%4 == 0`

(i) `bol9 = (b//2 > 0 and not(a//2 > 0)) or (not(b//2 > 9) and a//2 > 0)`

(j) `bol10 = not (bol3 and (a + b**2 > 50)) or d == e`

Hints: You can solve this exercise on the exercise sheet. If youâĂŹre struggling, try copying the relevant lines into Jupyter Notebook and execute them step by step.

Exercise 5.S01

For the initial values:

```
# Initialize variables
num1 = 10
num2 = 100
num3 = -10
```

and without copying the commands into Jupyter Notebook, evaluate the following expression:

```
bol = num1 > num2 and num2 > num3 or num3 > num1
```

Hints: Logical operators are easier to determine with brackets. Insert brackets in above expression and then evaluate one expression at a time starting with the innermost bracket until the entire command line is reduced to a Boolean.

Exercise 5.S02

For the initial values:

```
# Initialize variables
bol1 = True
bol2 = False
bol3 = True
```

and without copying the commands into Jupyter Notebook, evaluate the following expression:

```
bol = bol1 or bol2 or bol3 and not bol1
```

Hints: Logical operators are easier to determine with brackets. Insert brackets in above expression and then evaluate one expression at a time starting with the innermost bracket until the entire command line is reduced to a Boolean.

5.3 Difficulty: HARD

Exercise 5.H01

You can build all kinds of complex logical operators using combinations of `AND`, `OR` and `NOT`. In fact, you can even construct the `AND` operator using only `OR` and `NOT` and the `OR` operator using only `NOT` and `AND`.
In this exercise you wonâĂŹt have to write any code. Instead try and find a way to

(a) Rewrite `statement1 and statement2` using only `OR` and `NOT`

(b) Rewrite `statement1 or statement2` using only `NOT` and `AND`

Hints: You will need several instances of the operators in each case. Try starting with a fixed set of statements and try adding operators to obtain the result you want.

Chapter 6

IF and ELSE Statements

6.1 Difficulty: EASY

Exercise 6.E01

Write a code that accepts an arbitrary number and determines whether the number is

(a) positive (larger than zero)

(b) negative (smaller than zero)

(c) neither negative nor positive (zero)

Make sure the code displays your findings to the user.

Hints: You will need an `IF-ELIF-ELSE` chain structure for this exercise. The `IF` case could check whether the number is smaller than zero. The `ELIF` case could check whether the number is larger than 0 and the `ELSE` case handles numbers which are equal to zero.

Exercise 6.E02

Write a code that asks the user to enter one letter and then checks whether the letter is a vowel. The following letters are vowels: a, e, i, o, and u.

Hints: You can either use an `IF-ELIF-ELSE` chain statements to check whether the letter the user has entered is equal to any of the vowels or you could use a simple `IF-ELSE` statement with a more complex conditional test (using the `OR` operator).

Exercise 6.E03 [M]

Write a code that accepts an arbitrary number and determines whether the number is

(a) even

(b) odd

Hints: To determine whether a number is even or odd calculate the remainder of a division by two.The remainder of an even number divided by two is always zero. The remainder of an odd number divided by two is always 1. The remainder can be calculated using the modulus operator %.

Exercise 6.E04

Write a code that asks the user to enter their name and their nationality (eng, fr, de, esp, it, etc.). Then greet the user with a personalised message based on their first language:
English (eng): `"Good morning, [NAME]!"`
French (fr): `"Bonjour, [NAME]!"`
German (de): `"Guten Morgen, [NAME]!"`
Spanish (esp): `"Buenos dias, [NAME]!"`
Italien (it): `"Buongiorno, [NAME]!"`
where `[NAME]` is a place holder for the name the user has entered. Add more greetings in different languages if you know more! :)

Hints: Use an `IF-ELIF-ELSE` chain statement to check the nationality against the available options. Remember to use the == operator to check whether two strings are the same.

Exercise 6.E05

Write a code that acts as a virtual bouncer. The virtual bouncer should ask the user how old they are. If the user is 18 or older they are allowed to come in. If they are younger than 18, the virtual bouncer will ask them to leave.

Hints: When you accept user input the input will always be of primitive data type `string`. If you want to perform mathematical operations on the input you need to convert it into a number first. This can be done with the `int()` function.

6.2 Difficulty: MEDIUM

Exercise 6.M01

Write code that acts like a virtual ATM machine. At the beginning specify an account balance. Then allow the user to withdraw money. The ATM does not allow any overdrafts. If enough money is in the account, the ATM will carry out the transaction and display the new account balance. If not enough money is in the account the ATM will refuse to carry out the transaction.

Hints: Remember you will need to convert the input from string type to float type using the `float()` function. Check whether the account balance would be negative after the transaction. If so, the code should not allow the transaction.

Exercise 6.M02

Write a code that asks the user for the day of the week and the time (in full hours and 24h format) and determines whether a shop is currently open. The shopping hours are:
Monday to Saturday: 7am to 10pm
Sunday: 10am to 5pm

Hints: Remember you will need to convert the input from string type to integer type using the `int()` function. Using a nested IF-ELSE statement will reduce the amount of code you will have to write.

Exercise 6.M03 [M]

A year y is a leap year if it is exactly divisible by four unless itâĂŹs also exactly divisible by 100 and at the same time not exactly divisible by 400.
Ask the user for a year and then determine whether it is a leap year or not.

Hints: Remember you will need to convert input from string type to integer type using the `int()` function before performing mathematical operations. To check whether a number is exactly divisible by another number check whether the modulus % is zero.

Exercise 6.M04

Write a code that asks the user for a month and a day and returns the season. The seasons are determined as follows:
Spring: 21st of March to 20th of June
Summer: 21st of June to 20th of September

Autumn: 21st of September to 20th of December
Winter: 21st of December to 20th of March

Hints: Remember you will need to convert input from string type to integer type using the `int()` function before performing mathematical operations.

Exercise 6.M05

Write a code that asks the user to enter a string. Then determine whether the string is a palindrome. A palindrome is a word which reads the same forward or backwards. For example: "Anna" or "Madam".

Hints: If you want to reverse string `str` type `str[::-1]`. This will generate a substring from the original string from the beginning to the end with step -1. Meaning the string will be built from the end towards the beginning. Play around with the string reverse operation a bit before tackling the exercise. You might also find the `lower()` function useful when comparing the strings.

Exercise 6.M06

Write a code that accepts three numbers and

(a) returns the maximum

(b) returns the minimum

Hints: Use a nested `IF-ELSE` statement. Pick two numbers first and check which one is larger. Then compare this number to the third number to determine the overall largest number. Do the same for the minimum.

6.3 Difficulty: HARD

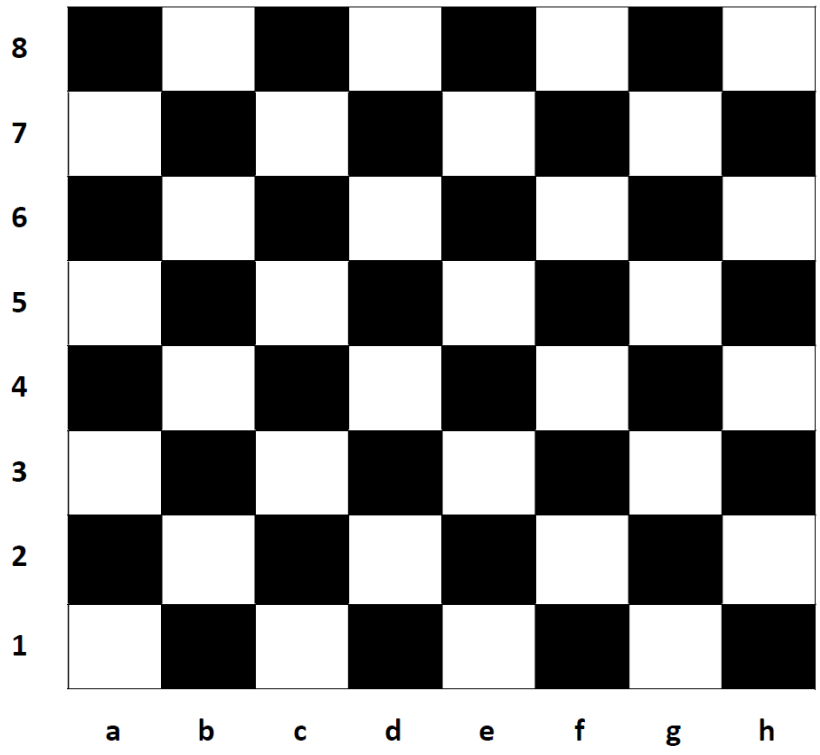
Exercise 6.H01

An Armstrong number (also known as narcissistic number) is a number that is the sum of its own digits each raised to the power of the digit number.
For example: $153 \rightarrow 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$
Write a code that asks the user to provide a three-digit integer. Then check whether the number is an Armstrong number. Display your result.

Hints: The four three-digit Armstrong numbers are: 153, 370, 371, 407. You will need to convert string characters into numbers using the `int()` function. To extract individual characters from a string use the corresponding string indices.

Exercise 6.H02

Consider the following chess board layout:



Every square on the board is identified by a number and a letter (for example: The lower left corner is identified by **1a**).
Write a code that asks the user for a square identifier and then determines whether the square in question is white or black.

Hints: Try to find a pattern before you start writing a code. Which conditions need to be met by the number part of the ID and which conditions need to be met by the letter part of the ID

for the square to be either black or white?
Remember: You can check whether a number is even or odd by calculating the modulus %. The modulus 2 of an even number is zero, the modulus 2 of an odd number is 1.

Chapter 7

Lists

7.1 Difficulty: EASY

Exercise 7.E01

Initialize lists with your five favourite animals/cars/sports/books/foods/whatever else you want. Display the content of the lists using the `print()` function.

Hints: When initializing a non-empty list, list objects are placed inside the square brackets and separated by commas.

Exercise 7.E02

Copy the following lists into your Jupyter Notebook:

```
1 citiesUK1 = ["London", "Birmingham", "Leeds", "Glasgow"]
2 citiesUK2 = ["Sheffield", "Bradford", "Mancheser", "Edinburgh"]
```

Generate a new list, `citiesUK`, containing the sum of `citiesUK1` and `citiesUK2` then (using list indices where appropriate)

(a) Print the entire list to your screen

(b) Print only the third object in the list to your screen

(c) Print the last object in the list to your screen

(d) Change the first object in the list from London to Cardiff then print your list again

(e) Generate a new list containing the cities Birmingham to Manchester using a list slicing operation. Display your new list using the `print()` function

Hints: You can use the + operator to add lists together. Every object in the list is assigned an index. You can access individual objects by calling their index in square brackets behind the list name: `myList[index]`. You can also generate sub-lists by declaring an index range in square brackets behind the list name: `myList[index1:index2]`.

Exercise 7.E03

Repeat Exercise 7.E01 but this time start by generating an empty list. Then, using the `append()` function, add your 5 favourite animals/cars/sports/books/foods/whatever else you want to the list. When youâĂŹre done display the content of the list.

Hints: The `append()` function will add new objects to the end of your list. Remember the full stop when applying a list operator to a list: `listName.FunctionName(arguments)`.

Exercise 7.E04

Copy the following list into your Jupyter Notebook:

```
1 # Initialize a list of numbers
2 numberList = [1, 40, -30, 100, 99, 0, 0, 3.14157]
```

Using the `insert(index, object)` function, keep adding objects to the list until the list content looks like this:

```
1 # Display list content
2 print(numberList)
```

```
[500, 1, 40, -30, -3.333, 100, 99, 0, 0, 20, 0, 3.14157, 0]
```

Hints: Remember that the index assigned to an object in a list might change if you insert a new object.

Exercise 7.E05

Copy the following lists into your Jupyter Notebook

```
# Initialize Lists
numberList = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]
stringList = ["February", "2018", "Europe", "Earth"]
```

and

(a) Use the `remove()` function to remove the object February from `stringList`

(b) Use the `del` function to remove the object Earth from `stringList`

(c) Use the `del` function to remove all number 3s from `numberList`

(d) Use the `remove()` function to remove all number 4s from `numberList`

Hints: The `del` function will remove an object by index, the `remove()` function by content!

Exercise 7.E06

Copy the following list into your Jupyter Notebook:

```
# Initialize Lists
babyNames2017 = ["Sophia", "Olivia", "Emma", "Ava",
                 "Isabella", "Mia", "Aria", "Riley",
                 "Zoe", "Amelia", "Jackson", "Liam",
                 "Noah", "Aiden", "Lucas", "Caden",
                 "Grayson", "Mason", "Elijah", "Logan"]
```

Ask the user for a name and then check whether the name is among the top 20 baby names of 2017 using the in function.

Hints: You can simply print the result of the `in` function to your screen or you can try including an `IF and ELSE` Statement that prints a message like `"[NAME] was/was not among the top 20 baby names of 2017"` where [NAME] is a place holder for the user input.

7.2 Difficulty: MEDIUM

Exercise 7.M01

List repetition allows you to generate larger lists more easily if objects are repeated within the list:

```
# Initialize a list containing the string "Hello"
# 4 times:
myList = 4 * ["Hello"]

# Display list content
print(myList)
```

```
['Hello', 'Hello', 'Hello', 'Hello']
```

Using repetition, generate a list that contains the number 1 once, the number 2 twice, the number 3 thrice, the number 4 four times, the number 5 five time, the number 6 six times, the number 7 seven times, the number 8 eight times, the number 9 nine time, and finally the number 10 ten times.

Hints: Remember, you can add lists together using the + operator.

Exercise 7.M02

Copy the following list initializations into your Jupyter Notebook:

```
# Initialize the lists
list1 = [i for i in range(1000) if i%3 == 0]
list2 = [i for i in range(1000) if i%4 == 0]
list3 = [i for i in range(1000) if i%5 == 0]
```

DonâĂŹt worry about understanding these lines. What you see above is called list comprehension which is not part of the syllabus but used here to quickly generate larger lists for you to work with. The commands will produce:

- list1 which stores all the numbers between 0 and 10,000 that are divisible by 3
- list2 which stores all the numbers between 0 and 10,000 that are divisible by 4
- list3 which stores all the numbers between 0 and 10,000 that are divisible by 5

Using the `len()` function, determine how many objects are contained in each of these lists.

Hints: Remember, the `len()` function is not a list operator but a function that works on multiple data types. ItâĂŹs therefore not initialized the way normal list operators are initialized.

Exercise 7.M03

Initialize the following list you Jupyter Notebook:

```
# Initialize the list
myList = 14 * [0] + [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

The list youâĂŹve just generated has quite a few 0 objects at the beginning. Instead of removing all the zeros, use the `index(0)` function generate a new list containing the none-zero elements starting with 1.

Hints: Use the `index()` function to find the index corresponding to 1, then generate a sub-list from that index to the end of the original list. You might find the `len()` function useful too.

Exercise 7.M04

In this exercise you will be putting together a shopping list. The initial list should contain the following items:

milk, eggs, red peppers, chicken, apples, onions, bread, coffee, and a cucumber

Using the list operators you have learned so far, perform the following actions on your list:

(a) Determine how many items are on your shopping list

(b) Print your shopping list in the following format:
```
Shopping list:
  - [ITEM]
  - [ITEM]
```
where [ITEM] is a place holder for your list objects.

(c) Add butter and spring onions to your list

(d) Add lemons to your list but insert them right before the onions because they are right next to each other in the shop so you can pick them up in one go

(e) Remove the coffee from your shopping lists as youâĂŹve just discovered a new pack in the cupboard

(f) Using the in function make sure that apples are on your shopping list

(g) Finally print your updated list again in the same format as before

Hints: You will need the `len()` function, the `print()` function, the `append()` function, the `insert()` function, the `index()` function, the `remove()` function, and the `in` function. Remember, you can generate tabs inside a string with `\t`.

Exercise 7.M05

Copy the following list initialization into your Jupyter Notebook:

```
# Initialize the list
numberList = [1 if i%11==0 else 0 for i in range(10000)]
```

DonâĂŹt worry about understanding this line. What you see above is called list comprehension which is not part of the syllabus but used here to quickly generate larger lists for you to work with. The command will produce:

- `numberList` which stores, for every number between 0 and 10,000, whether the number is divisible by eleven. If the number is divisible 11 eleven, the value in the list will be one, if the number is not divisible by 11 the number will be 0.

Using the `count()` function, determine how many numbers between 0 and 10,000 are divisible by 11.

Hints: Divisibility by 11 corresponds to a list element 1. So you will need to count how many times the number one is contained in the list.

Exercise 7.M06

Copy the following list into your Jupyter Notebook:

```
# Initialize a list
numberList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Using the `pop()` function, write a program that stores the reverse of `numberList` in a new list `numberListReverse`.

Hints: Start with an empty list `numberListReverse` and fill it using the `pop()` function on `numberList` multiple times.

Exercise 7.S01

Start by initializing an empty list. Then ask the user for three cities and append them to the empty list. Finally check whether London was one of the cities the user entered.

Hints: You can initialize empty lists by simply leaving out the objects between the square brackets. You can then either check each list object manually to test whether one of the strings is "London" or you can use the `in` function.

Exercise 7.S02

Start with the following list of the London boroughs:

```
# Initialize the list
londonBoroughs = ["Barking and Dagenham", "Merton",
                  "Hammersmith and Fulham", "Kensington and Chelsea",
                  "Tower Hamlets", "City of London", "Bexley",
                  "Havering", "Richmond upon Thames", "Croydon",
                  "Waltham Forest", "Hillingdon", "Greenwich",
                  "Bromley", "Barnet", "Redbridge",
                  "Kingston upon Thames", "Southwark", "Wandsworth",
                  "Haringey", "Brent", "Ealing", "Hounslow",
                  "Lewisham", "Westminster", "Islington", "Harrow",
                  "Hackney", "Camden", "Enfield", "Lambeth", "Sutton",
                  "Newham"]
```

And bring it into alphabetical order.

Hints: Use Google to figure out whether there is a list operator that can do this for you.

7.3 Difficulty: HARD

Exercise 7.H01

In your Jupyter Notebook, initialize the following 2-dimensional list:

```
# Initialize 2D list
myNestedList = [[0], [1, 2, 3], [4, 5, 6, 7], [8, 9, 10]]
```

This parent list contains 4 child lists of varying length. Make sure you understand the structure of this nested list before continuing (sometimes it helps drawing list structures on a piece of paper).
Once you feel like you understand the generated list, continue to the exercises:

(a) Using list indices print the first, then the second, then the third, and finally the fourth child list

(b) Using list indices, print out elements 0, 3, 4, and 9

(c) Replace the object 8 with 88

(d) Generate a sub-list containing the elements 4 to 6 and 9 to 10 using list slicing methods

(e) Append the new child list [11, 12, 13, 14, 15, 16] to the end of the nested list

(f) Insert the new child list [-2, -1] at index 0 in the nested list

(g) Using the `del` function, remove the second child list in the nested list

(h) Using the `remove()` function, remove the elements 4 and 9 from the nested list

(i) Using the `in` function, check whether the number 10 is in any of the child lists

(j) Using the `len()` function, determine how many child lists are contained in the parent list

(k) Using the `len()` function, determine the length of the individual child lists

Chapter 8

Loops

8.1 Difficulty: EASY

Exercise 8.E01

Using a `WHILE` loop, print the string `"Hello World"` 50 times.

Hints: DonâĂŹt forget to include a counter in this `WHILE` loop.

Exercise 8.E02

Copy the following list into your Jupyter Notebook:

```
# Initialize a list
numberList = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Using a `FOR` loop, print all the elements in the list.

Hints: `FOR` loops donâĂŹt need a counter. The loop variable will automatically cycle through all elements in a sequence

Exercise 8.E03

Using a `WHILE` loop, generate a list containing all the numbers between 0 and 1,000.

Hints: You can solve this exercise with or without a counter. Remember: You can append new elements to a list using the `append()` function. If you want to try the loop without the counter, remember you can determine the length of a list using the `len()` function.

Exercise 8.E04

Using a `WHILE` loop, ask the user for their five favourite movies and store them in a list called `favMovies`.

Hints: Use the `input()` function to request user input and the `append()` function to add new elements to a list.

Exercise 8.E05

Using a `WHILE` loop, generate a list that stores the squares of all numbers between 0 and 1,000.

*Hints: You can calculate the square of a number using the ** operator.*

Exercise 8.E06

Copy the following list into your Jupyter Notebook:

```
# Initialize a list
numberList = [0.001, 45, 234, 0.45, 2, 94, 63, -0.5]
```

Using a `FOR` loop, generate a new list containing every object in the original list multiplied by 10.

*Hints: Use the * operator to perform a multiplication.*

8.2 Difficulty: MEDIUM

Exercise 8.M01

Using a `WHILE` loop, generate a list containing all the numbers between 1 and 10,000. Then, using another loop (`WHILE` or `FOR`) calculate the sum of all the numbers in the list.
Bonus: Calculate the average of all the numbers in the list.

Hints: You can use list indices to access specific objects in your list.

Exercise 8.M02

Using a `WHILE` loop generate a list of all numbers between 0 and 1,000,000 that are a multiple of 17. A number is a multiple of 17 if the modulus 17 of the number is equal to zero meaning:

```
number % 17 == 0
```

Determine how many objects are in the list.

Hints: Insert an IF and ELSE statement inside your loop structure.

Exercise 8.M03

Using the `range()` function and a `FOR` loop print all the numbers between 0 and 1,000.

Hints: You can initialize the range instead of the usual sequence in the initialization line of your `FOR` loop.

Exercise 8.M04 [M]

Write a code that accepts a number x from the user and calculates the factorial of x. The factorial $x!$ of x is given by:

$$x! = x \cdot (x-1) \cdot (x-2) \cdot ... \cdot 2 \cdot 1.$$

Hints: Your counter doesn't necessarily have to count up, it can also count down.

Exercise 8.M05

Initialize a list containing all numbers between 1 and 20. Using a loop structure reverse the list.

Hints: Store the reverse list in a new list and use the `pop()` function.

Exercise 8.M06

Write a code that asks the user for a city and stores the city in a list. Continue asking the user for another city until the user enters the string `STOP` instead of a new city.

Hints: You will need to check the input at every iteration step to test whether the `STOP` command has been entered. Use the `BREAK` command to forcefully exit a loop.

Exercise 8.M07

Using a `WHILE` loop and the `CONTINUE` command, write a code that generates a list of all the numbers between 0 and 1,000 but do not include any numbers that are divisible by 5. Count the number of elements in your list once youâĂŹre done.

Hints: To check whether a number is divisible by 5, check whether the modulus 5 is equal to zero. If `num % 5 == 0` then the number is divisible by 5.

Exercise 8.M08

In this exercise you will pick a number x and allow the user to guess the number.

(a) Set x to a number between 0 and 10. The user is allowed to guess three times. If the user guesses the number, the user wins. If the user fails to guess the number after three tries, the computer wins. Display a message declaring the winner.

(b) Set x to a number between 0 and 1,000. Allow the user to guess until they have guessed the number correctly. Whenever the user enters a wrong number, tell them whether the actual number is smaller or larger than the guess. Keep track of how many guesses it took the user to get to the right number.

Hints: Remember you need to convert the user input from a string to an integer using the int() function to perform mathematical operations on it. You will need `IF and ELSE` statements inside your loop (maybe even nested `IF and ELSE` statements).

Exercise 8.M09 [M]

A prime number is an integer that can be divided without a remainder only by 1 or itself.

(a) Write a code that stores all prime numbers between 1 and 1000 in a list. (Note: 1 is not a prime number!)

(b) Count the number of prime numbers between 1 and 1000 without using the len() function.

Hints: You can check all prime numbers between 2 and 10,4729 here: `https://primes.utm.edu/lists/small/10000.txt`. *You will need a nested loop and* `IF and ELSE` *statements.*

Exercise 8.M10

Ask the user to enter a three-digit pin code containing numbers between 0 and 9. The write a code that guesses the pin code.

Hints: You will three nested loops each with their own counter, the `BREAK` and the `CONTINUE` command. Remember: To convert strings into integers use the `int()` function, to convert integers into strings use the `str()` function.

Exercise 8.M11

You have ten students in your class:
Oliver, Harry, George, Jack, Jacob, Olivia, Amelia, Emily, Isla, and Ava.
You ask them to pair up for an exercise. Using nested loops, find all possible group constellations. Count the total number of constellations.

Hints: You will need nested loops with their own individual counters here. Remember that you canâĂŹt pair students with themselves so you will also need an `IF and ELSE` *statement structure to take care of those cases.*

Exercise 8.M12 [M]

The number π is a mathematical constant you can access via python libraries. You can also calculate it using the Leibniz formula:

$$\frac{\pi}{4} = \sum_{k=0}^{n} \frac{(-1)^k}{2 \cdot k+1}\Big|_{n\to\infty}.$$

We cannot solve this equation for all values of k, but by choosing a large enough n when can approximate the value of π using a loop over k for the summation:

$$\pi \approx 4 \cdot \sum_{k=0}^{n} \frac{(-1)^k}{2 \cdot k + 1}.$$

Write a code that approximates π for

(a) $n = 10$

(b) $= 100$

(c) $= 1000$

and calculate how much the obtain values deviate from the `math.pi` value.

Hints: To use the `math` library you need to import it first using `import math`.

Exercise 8.M13

Generate two lists:

```
L1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

and

```
L2 = [8, 9, 10, 11, 12, 13]
```

and write a code that

(a) compares both lists and returns a new list containing elements that are present in both `L1` and `L2`.

(b) compares both lists and returns a new list containing elements that are present in either `L1` or `L2` but not in both.

Hints: You can test whether an object is in a list using the `in` function.

Exercise 8.M14

Write a code that accepts two strings from the user. Check whether the two strings are anagrams of each other. Display the result.
`string1` and `string2` are anagrams if `string1` can be turned into `string2` simply by reshuffling the letters in it. For example: drawer and reward are anagrams.

Hints: You can check your code with this website: `https://www.wordplays.com/anagram-solver/`.

Exercise 8.S01

Write a code that finds the longest string in a string list of arbitrary length.

Hints: You can find the length of a list using the len() function or use a FOR loop to iterate over all elements in a list. You can generate random words here: `https://www.randomlists.com/random-words` to test your code.

Exercise 8.S02

Remember Armstrong numbers from exercise 6.H01:
An Armstrong number (also known as narcissistic number) is a number that is the sum of its own digits each raised to the power of the digit number.
For example: $153 \rightarrow 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$
Instead of writing a code that tests whether a 3-digit number is an Armstrong number as we did in 6.H01, write a code that take a number of arbitrary length and determine whether it is an Armstrong number.

Hints: You can check your code using this website: `http://mathworld.wolfram.com/NarcissisticNumber.html`.

8.3 Difficulty: HARD

Exercise 8.H01

Generate a two-dimensional list with each element of the parent list containing a child list which in turn contains two elements: The name of a costumer and their mobile phone number. For example:

```
costumer = [[Donald Duck, 123456789], [Mickey Mouse, 987654321], ...]
```

Write a code that asks the user for the name of a costumer. If the name is not in the list, print an error message. If the name is in the list, return the corresponding mobile phone number.

Hints: When youâĂŹre working with nested lists, remember that each child list will correspond to an additional index. Sometimes it helps drawing the list structure on a piece of paper.

Exercise 8.H02 [M]

There are multiple ways to express numbers. The most common one is the decimal number system (base 10) which we use in our everyday life. Computers on the other hand tend to use a binary system (base 2) in which numbers are represented by combinations of the digits 0 and 1. In this exercise you will

(a) Write a code that converts any decimal integer provided by the user into a binary number

(b) Write a code that converts any binary number provided by the user and converts it into a decimal integer

Below you can find an explanation and worked examples on how the conversions work. Visit `http://calc.50x.eu/` to check whether your code works.

Converting a Decimal Number into a Binary: To convert a decimal number x into a binary perform the following steps:

1. Divide x by 2
2. Take note of the remainder $x\%2$ (which will be either 0 or 1)
3. Set x equal to the result of the division: $x = x/2$
4. Repeat step 1-4 until $x = 0$
5. Reverse the order of the remainders

Example:
LetâĂŹs convert the number 108_2 into a binary:

$x = 108_{10}$			
$x/2 =$	54	Remainder =	0
$x = 54$			
$x/2=$	27	Remainder =	0
$x = 27$			
$x/2 =$	13	Remainder =	1
$x = 13$			
$x/2 =$	6	Remainder =	1
$x = 6$			
$x/2 =$	3	Remainder =	0
$x = 3$			
$x/2 =$	1	Remainder =	1
$x = 1$			
$x/2 =$	0	Remainder =	1
$x = 0$			
1101100_2			

Converting a Binary Number into a Decimal: To convert a binary number b into a decimal perform the following steps:

1. Write down the binary number
2. Underneath each binary digit note down the powers of two going from right to left
3. Multiply each binary digit with its corresponding power of two
4. Add the power of two values together

Example:
LetâĂŹs convert the number 1101100_2 back into a decimal:

Binary:	**1**	**1**	**0**	**1**	**1**	**0**	**0**
Power of two:	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
	↓	↓	↓	↓	↓	↓	↓
Multiply:	$64 \cdot 1 = 64$	$32 \cdot 1 = 32$	$16 \cdot 0 = 0$	$8 \cdot 1 = 8$	$4 \cdot 1 = 4$	$2 \cdot 0 = 0$	$1 \cdot 0 = 0$
Decimal:	$64 + 32 + 8 + 4 = 108_{10}$						

*Hints: You will need the modulus operator %, the floor division operator //, the exponent operator **, and eventually you will want to reverse a string which can be done via* `str[::-1]`.

Exercise 8.H03 [M]

In this exercise you will be writing a code to perform matrix additions and multiplications. A matrix is a mathematical construct with numbers arranged in columns (c) and rows (r). The simplest example is the vector with 1 column and an arbitrary number of rows ($1 \times r$), this is called a vector.
In this exercise however, we will be looking at larger matrices. For these objects, addition and multiplication are bit a bit more complex. Below you can find the addition and multiplication rules for ($n \times n$) matrices:

Matrix-Number Multiplication:

$$4 \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 4 \cdot 5 & 4 \cdot 6 \\ 4 \cdot 7 & 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 20 & 24 \\ 28 & 32 \end{bmatrix}$$

Matrix Addition:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Matrix-Matrix Multiplication:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Write a code that

(a) multiplies a ($c \times r$) matrix by a number.

(b) calculates the sum of two matrices. Start with the simple $(2 \times 2) \times (2 \times 2)$ case as shown in the examples above. Then extend it to the more general case of $(n \times n) \times (n \times n)$.

(c) calculates the product of two matrices. Start with the simple $(2 \times 2) \times (2 \times 2)$ case as shown in the examples above. Then extend it to the more general case of $(n \times n) \times (n \times n)$.

Hints: To help you visualize larger matrices, I have included a python script called `matrixOperations.py`. *It's in the same directory as the solution to this exercise. Place the file in the same folder as your Jupyter Notebook or script. The add* `from matrix_operations import matrixPrint` *to the top of your script. You can now use (a very basic) matrices visualization routine by typing* `matrixPrint(matrixName)`.

Chapter 9

File Operations

9.1 Difficulty: EASY

Exercise 9.E01

Write a code that generates a file `E09.E01_HelloWorld.txt` and write the following strings to it:

```
Hello World
Goodbye World
```

Hints: You will need the `open()`, `write()`, and `close()` functions. Remember: You can generate new lines by including the string formatter `\n` in the string. To write to a file, use the keyword `w`.

Exercise 9.E02

Write a code that accepts a food item from the user and appends it to a file called `E09.E02_ShoppingList.txt`. Run your code multiple times to generate a longer shopping list.

Hints: You will need the `open()`, `write()`, and `close()` functions. Remember: You can generate new lines by including the string formatter `\n` in the string. To append to a file, use the keyword `a`.

Exercise 9.E03

Write a code that opens the file `E09.E03_LotROpeningLine.txt` and prints the content to the screen.

Hints: You will need the `open()`, `print ()`, and `close()` functions. To read from a file, use the keyword `r`.

Exercise 9.E04

Using a `WHILE loop`, write all the numbers between 0 and 1,000 into a file called `E09.E04_OneThousand.txt`. Each number should be in a new line.

Hints: Remember to update the counter inside the `WHILE` loop and generate new lines using `\t`. Remember to convert numbers into strings using the `str()` function.

Exercise 9.E05

Write a code that opens the file `E09.E04_OneThousand.txt` which you created in exercise 9.E04. Read in the information in the file and calculate the sum of all numbers contained.

Hints: You will need to convert each line into an integer using the `int()` function.

Exercise 9.E06

Open the file `E09.E06_PiToOneMillionDigits.txt` which contains the number π to one million digits. Write a code that checks whether your birthday (in the following format `ddmmyyyy`, for example: `01011960` for 01.01.2016) is contained in the first million digits of π.

Hints: Before you read in the file, have a look at it first using a text editor. Note how the file contains only one line. DonâĂŹt convert either the line or your birthday in a number. Use strings instead.

Exercise 9.E07

Open the file `E09.E07_PrideAndPrejudice.txt` (in the public domain and available from `https://www.gutenberg.org`). Read in all the lines in the file and append each to one big string. Then count the number of words in the string and therefore the number of characters in Jane AustenâĂŹs Pride and Prejudice.

Hints: You can add strings to other strings via the + operator. You can count the number of characters in a string using the `len()` function.

9.2 Difficulty: MEDIUM

Exercise 9.M01

Repeat exercise 8.M06 and write a code that asks the user for a city inside an infinite `WHILE` loop until the user enters the string `STOP` instead of a new city. This time however, instead of storing the individual user inputs in a list, write them to a file `E09.M01_Cities.txt`.

Hints: Set the loop condition to `TRUE` to generate an infinite loop. Remember: You can exit a loop prematurely by triggering the `BREAK` command.

Exercise 9.M02

Some data files will come with comment lines at the top explaining the content of the file. They usually begin with a #. ItâĂŹs always a good idea to check your file before trying to open it so you know exactly what it looks like. If the file includes a comment line, you will want to ignore it while reading in the data.
Write a code that opens file `E09.M02_IrishProvinces.txt` and prints the Irish provinces stored inside but not the comment line!

Hints: There are several ways to check whether a line is a comment line. You could check whether the line includes a #, you could check whether the first character in the line is equal to #, or you could use the `startswith()` string operator.

Exercise 9.M03

Sometimes you get data files that includes empty lines or lines containing nothing but spaces. If you donâĂŹt sort these out, you will end up with a lot of useless overhead in your data lists. Write a code that opens the file `E09.M03_USStates.txt`, extract all the US states in the file and store them in a list called `statesUS`. Make sure to ignore lines that are empty or contain nothing but strings. Remember to remove the new-line delimiter as well!
Once youâĂŹre done, display the content of your list and count the number of states.

Hints: You can remove all spaces from a string using the `strip()` function. If what remains is an empty string the original string contained nothing but spaces to begin with.

Exercise 9.M04

Write a code that opens the file `E09.M04_WorldCountryCapitals.txt`. This file contains two data columns. The first column contains all the countries in the world, the second column

the corresponding capitals. The columns are separated by a tab. Extract the data from the file and store the countries in a list called countries and the capitals in a list called capitals. Remember to remove the new line formator `\n` if necessary.
Bonus: Add a routine that asks the user for a country and returns the capital.

Hints: A tab is represented by `\t` in Python and strings can be split using the `split()` function. You can remove the last character in a string by putting writing `str = str[:-1]` at the end.

Exercise 9.M05

Write a code that accepts costumer feedback and stores it in a file called `E09.M05_Reviews`. The feedback should include: The name of the costumer, the purchase date, the bought product, and the feedback itself.
Bonus: Add another piece of code that accepts a name and returns the feedback given by the costumer.

Hints: Store the costumer feedback in columns, then use the `split()` function to separate the data when you read it in.

Exercise 9.M06

Have a look at file `E09.M06_DeliveryRefernceNumbers.txt`. The file contains parcel reference numbers in order of delivery. It is assumed that each delivery takes between 5 and 15 minutes (this includes the drive to the address and the delivery itself). The delivery driver starts work at 9 am.
Write a code that accepts a reference number from the user and returns an estimated delivery time slot. Make sure you can handle invalid delivery references too.

Hints: You will need to calculate how much time it will take to deliver the parcels before the delivery in question. If all deliveries take five minutes you get the earliest estimate. If all deliveries take 15 minutes you get the latest estimate.

Exercise 9.M07

Have a look at file `E09.M07_Employees.txt`. The file contains a list of employees with their name, ID number and birthday. However: Some entries are incomplete and are missing the birthday.
Write a code that reads in the data in the file and:

(a) stores the name, ID and Birthday of every employee with a complete set of information in a new file `E09.M07_EmployeesComplete.txt`.

(b) stores the names and ID of every employee with a missing birthday in a new file `E09.M07_EmployeesMissingBDay.txt`.

(c) stores the name, ID and birthday of every employee above the age of 65 in a new file `E09.M07_EmployeesRetire.txt`.

(d) accepts a date from the user in the format `dd.mm` and prints a list of employees whose birthday is on that day with their age in brackets. You can assume all input dates are valid.

(e) Have a look at the `datetime` library (`https://docs.python.org/2/library/datetime.html`) and write a code that automatically lists all employees with birthdays on the day the code is being executed. You can use the `datetime.date.today()` function to find todayâĂŹs date.

(f) You want to send a letter to every employee on their birthday. Generate a loop that writes a personalised birthday message into a file called `E09.M07_BDWishes-[LastName][FirstName].txt` where `[LastName]` and `[FirstName]` are place holders for the last and first name of the employee.

Hints: You can solve (a) to (c) inside one big loop.

Exercise 9.M08

Write a code that reads in the DNA sequence stored in `E09.M08_DNA-Murderer.dat` and compare it to the suspect's DNA sequences stored in `E09.M08_DNA-Suspect0000.dat` to `E09.M08_DNA-Suspect0050.dat` to figure out who's the murderer.

Hints: Use a loop to quickly read in all suspect files. Remember the comparison operator ==.

Exercise 9.M09

The file `E09.M09_WorldPopulation.txt` contains the world population from 10,000 years ago until 2015 (source: `https://ourworldindata.org/world-population-growth`). The first line corresponds to the year, the second column to the number of human inhabitants. Read in the file and determine the year in which the world population surpassed 1 billion (1,000,000,000).

Hints: Remember to ignore comment lines and convert strings into numbers before performing mathematical operations on them.

Exercise 9.S01

Initialize two lists. The first list `numList` should contain all the numbers between 0 and 1,000. Using the first list, generate a second list `squareNumList` containing the square roots of all them numbers in `numList`.
Once you have created both lists, create a file called `E09.S01_SquareRoots.txt` and write the content of the lists to it so the file ends up looking like this:

E09.S01_SquareRoots.txt

```
# Square Roots of Numbers Between 0 and 1,000
# num    sqrt(num)
0    0.0
1    1.0
2    1.4142135623730951
3    1.7320508075688772
4    2.0
5    2.23606797749979
6    2.449489742783178
7    2.6457513110645907
8    2.8284271247461903
```

Hints: Remember to include \n and \t to format your output.

Exercise 9.S02

Open the file `E09.S02_SN2005ek.txt` which contains spectral data of a supernova explosion (Source: `https://wiserep.weizmann.ac.il/`).
Write a code that reads in the information from the file. The first column corresponds to wavelength, the second column corresponds to flux density. Find the maximum flux and the corresponding wavelength.

Hints: You can use the `split()` function to break down a string into smaller strings which are returned in a list. Remember to convert strings into numbers first using the `float()` function when you want to perform a mathematical operation on them.

9.3 Difficulty: HARD

Exercise 9.H01

In this exercise you will be generating a Hangman game.
The game works as follows:
The computer will choose a word at random. You can then start guessing letters. Once you have guessed all letters in the word you have won. If the computer finishes drawing the hangman first, you lose.
To write the Hangman game:

1. Download the file `E09.H01_sowpods.txt` (`https://github.com/jesstess/Scrabble/blob/master/scrabble/sowpods.txt`)into your local working directory. This file contains about 25.000 words commonly used in scrabble.
2. Read the file and choose one random word and display an output like this: `"I'm thinking of the following word: _ _ _ _ _ _ _ _"` where the number of underscores is equivalent to the number of letters in the word. To choose a random word, import the `random` library via `import random` then generate a random integer between 0 and the length of the list `wordList` in which you have stored all the words from `E09.H01_sowpods.txt`: `randomNumber = int(random.random() * len(wordList))`
3. Prompt the user to guess a letter (you can assume the input is always valid). If the letter is not in the word, prompt the user to guess again and increase the number of wrong guesses by one. If the letter is in the word (e.g. the letter a), update the output like this: `"I'm thinking of the following word: _ _ _ a _ _ a _"` and prompt the user to guess the next letter.
4. If the user has guessed the letter before (whether it was correct or not), tell them to guess again but do not increase the number of incorrect guesses.
5. If the user has guessed the entire word, display it and offer congratulations
6. Finally, keep track of the number of incorrect guesses. After x (for example $x = 11$) incorrect guesses, the user loses the game. Tell the user how many guesses they have left after each guess.

Bonus:

1. Instead of telling the user how many guesses they have left, display a hangman figure after each wrong guess.
2. Generate a wrapper that runs the game again if the user enters Y after a round and stops if the user enters N

*Hints: The solutions to this exercise won't include functions since they are not covered in the course these exercises were designed for. It is **much** easier to solve this exercise with functions, so please use them if you know how to!*

Chapter 10

Bonus: Functions

10.1 Difficulty: MEDIUM

Exercise 10.01

Generate a function `PrintMessage()` that prints a message of your choosing to the screen when called.

Hints: Remember to include the def and the : when initializing a function.

Exercise 10.02

Generate a function `PrintUserInput(userInput)` that accepts one argument (for example a user input) and prints it to the screen.

Hints: Use the `input()` function to request user input.

Exercise 10.03

Generate a function `CalculateSum(a, b, c, d)` that takes four arguments `a`, `b`, `c`, and `d` (either integers or floats) and returns the sum `a + b + c + d`.

Hints: DonâĂŹt forget the return keyword if you want your function to return a value to the code.

Exercise 10.04

Generate a function `CalculateAge(birthyear)` that takes a persons birth year and returns their current age.

Hints: If you want to include user input, remember to convert strings to integers before performing any mathematical operations on it.

Exercise 10.05

Generate a function `FindMaximumInList(list)` that accepts a list of any length and returns the maximum number in the list.
Bonus:
Generate a function `FindMinimumInList(list)` that accepts a list of any length and returns the minimum number in the list.

Hints: Remember you can iterate over a list with either a for or a while loop.

Chapter 11

Bonus: Classes

11.1 Difficulty: MEDIUM

Exercise 11.01

Generate a class `Dog`. The class should contain the attributes name, age and breed. Include a function `DisplayInformation()` that displays the object information like this: `"[Name] is [YEARS] years old and a [BREED]"`, where `[Name]`, `[YEARS]` and `[BREED]` are place holders.

Hints: Remember the `__init__()` method as well as the getters and setters.

Exercise 11.02

Generate a class `Vehicle`. The class contains the attribute `value`. Then generate the child classes: `Car` and `Bicycle`. The `Car` child class contains the attribute `brand`, the `Bicycle` child class contains the attribute `colour`.

Hints: Remember the `__init__()` method as well as the getters and setters. for new attributes in the child classes.

Exercise 11.03 [M]

A complex number can be expressed as

$$a + i \cdot b$$

where a and b are real numbers and i is the imaginary number which solves $i^2 = -1$.
Generate a class `Complex`. The class contains the attributes `real` and `imaginary`. The real attribute corresponds to a, the imaginary attribute corresponds to b.

Add a function `DisplayImaginaryNumber()` that prints your number in the following format: `a + i * b` where `a` and `b` are place holder.
Add a function `AddImaginaryNumbers()` that adds two `Complex` objects together. To add two Complex objects together do:

$$(a + i \cdot b) + (c + i \cdot d) = (a + c) + i \cdot (b + d)$$

Add a function `SubtractImaginaryNumbers()` that subtracts two `Complex` objects from each other. To subtract two `Complex` objects do:

$$(a + i \cdot b) - (c + i \cdot d) = (a - c) + i \cdot (b - d)$$

Hints: This exercise requires more advanced mathematical knowledge, skip it if youâĂŹre not comfortable with the concepts shown here.