Babraham Bioinformatics

Introduction to Python

Exercises

Version 2020-08

Licence

Chapter 1: Getting started with Python

Exercise 1: introducing Thonny

1a
Open the program Thonny and browse the menus and options.

1b
What version of Python are you running in Thonny?

1c
Go to the Thonny homepage at https://thonny.org and briefly familiarise yourself with this page.

1d
Run the Hello World! program in the Thonny interactive Window.

1e
Modify the Hello World! program to print to the screen “Saluton Mondo!”

Chapter 2: Data types and expressions

Exercise 2: introducing expressions

2a
Use the Thonny interactive window to perform the following numerical calculations:

- i) Five plus seven (5 + 7)
- ii) Eight minus ten
- iii) Nine multiplied by eleven
- iv) Minus fifteen divided by three
- v) Three cubed

2b
Use the Thonny interactive window to perform the following numerical calculations:

- i) Calculate the square root of 2 (remember: determining the n^{th} root of a number is the same as raising that number to the power of 1/n)
- ii) One trillion is 1,000,000,000,000. Multiply 3.2 trillion by 2.5. To simplify this calculation, use scientific notation to represent a trillion.
- iii) Determine the value in grams of 2.3 picograms multiplied by 45.3. (Search the internet for an explanation of the term pico, if you don’t know what this means).
- iv) What is the remainder of dividing nine by two?
- v) Is 1522756 exactly divisible by 37?
- vi) What is the floor division of 22 divided by 7?

2c
Use the Thonny interactive window to write **one-line** expressions to calculate the following (include all the values mentioned). Use parentheses to make the calculations more readable, and then compare the results after removing these brackets.

- i) I buy 5 sandwiches at £1.20 and 8 rolls at £2.15. How much will they all cost?

ii) One thousand pounds is shared equally between 20 men and 30 women. How much does each person receive?

iii) I work for 3 hours at £15.20 per hour, 9 hours at £17.45 per hour and 2 hours at £24.00 per hour. I then pay 20% income tax on the money earned. How much money do I take home?

Exercise 3: working with strings

3a

Use the Thonny interactive window to create the following strings:

i) Once upon a time

ii) Ready…
Steady…
Go!

Do ii) with triple quotes and then using the \n metacharacter.

iii) Toto, I've got a feeling we're not in Kansas anymore.

Do iii) with using contrasting single/double quote pairs. Also do this by using the backslash character.

iv) Concatenate the following strings into a single one-line string:
Con
cat
en
ate

v) Concatenate the following words into the phrase “To be or not to be” from its constituent words (think about how to create the spaces between words).

vi) Write a **single-line** expression that comprises following phrase 20 times: "All work and no play makes Jack a dull boy".

3b

i) The well-known phrase says that “There is no I in team”. Demonstrate that this is indeed true using the Python `in` operator. Try using `not in` as well. What happens?

ii) Does the following nucleotide sequence contain a BglII restriction site (AGATCT)?
GATATCTAGTCTTCTGATAGAGATCTGATGGGGATTATTATAGCTTCTGATCGGTTT

3c

Use subscription and splicing to extract the specified text from the phrase:
“Mighty oaks from little acorns grow”

i) Extract the first five characters

ii) Extract the word oak

iii) Extract the last four characters (using a negative value)

iv) Return alternate characters (i.e. M, g, t,….)

v) Reverse the text

Exercise 4: Boolean logic

4a

Use the Thonny interactive window to write **one-line** expressions to do the following comparisons. Take note of the answer (this should be a Boolean value).

i) Evaluate whether 0 is **equal** to -0
ii) Evaluate whether 10 is **not equal** to 10.000
iii) Evaluate whether (856 / 7) is **greater** than (864 * 7.2)
iv) Is 9/3 **less than or equal to** 33/11?

4b

Evaluate the following in the Thonny interactive window. Think about what the result for each evaluation means i.e. the logic behind the results. What happens if `None` is returned?

i) `True and True`
ii) `True and False`
iii) `False or True`
iv) `True and None`
v) `None or True`
vi) `1 or 5`
vii) `0 or 0 or False or 1`
viii) `not False`
ix) `not 40`

Chapter 3: names, functions and methods

Exercise 5: writing your first script

5a

i) Use the scripting area of Thonny to write a script that prints out “Hello World! This is my first script.”
ii) Save the script as hello_world.py
iii) Run the script and you should see your message print to the screen.
iv) Create another script using Python code you have learnt already on this course. Try to write at least 5 lines of code then run the script.

Exercise 6: introducing names

6a

i) Remember Pythagoras’ theorem which is used to calculate the length of the hypotenuse of a right-angled triangle? The theorem may be represented algebraically as $c^2 = a^2 + b^2$ (where c is the length of the hypotenuse and a and b represent the lengths of the other two sides).
Write a script called `pythagoras.py` in which a=3, b=4. The script should then calculate c and print its corresponding value to the screen.
ii) Modify the script you created in i) to have different values for a and b. The value printed to the screen should now change. Does this happen?

6b

i) Write a script called `say_my_name.py` to print your name to the screen 50 times (on a different line each time). Assign your name to the variable `my_name`.

ii) Let's now create a `reverse_my_name.py` script. Create a new variable called `reverse_name` (which will be the reverse of your name - i.e. the letters in reverse order). The script should be able to calculate this value automatically from the value of `my_name`. Print the `reverse_name` to the screen 20 times.

iii) Let's now write a script called `say_my_full_name.py` that has variables `first_name`, `middle_name` and `last_name`. Assign your name to these variables (make up a middle name if you don't have one).
Now create a variable called `full_name` which comprises: first name, middle initial and last name e.g.
Homer J Simpson
The variable should contain spaces to separate values. This variable should be generated by your code from `first_name`, `middle_name` and `last_name` variables. Print the `full_name` variable to the screen.

Exercise 7: f-strings

7a

Let's create a new script called `f_string.py`. Declare the variables `first_name` and `last_name` and assign them your names.

i) Create an f-string called `greeting` that says hello to you and then includes your first name and last name. Print `greeting` to the screen.

ii) Use a string subscription to calculate your initials from **within** an f-string expression. Print the results.

iii) Repeat the previous step, but right align the initials (using a fixed-width line of 45 characters).

iv) In the same script, use an f-string to format the square root of 2, to 5 decimal places and then prints the result to the screen.

Exercise 8: built-in functions

8a

Refer to the list of functions in the course text. Now use the `help()` function to identify the functions that perform the tasks listed below. When you have found this out, think how each of these functions might be useful in your code.

i) Return the absolute value of a number
ii) Round a number
iii) Takes command line user input
iv) Checks whether an object is some kind of function (Hint: methods and functions are both kinds of what?)

8b

In the interactive window use the appropriate function to calculate:

i) The maximum value of 1, 4, 45, 9, -3, 0.3
ii) The minimum value of the same numbers
iii) The maximum value of the characters: 'A', 'B', 'z','@'
iv) The length of the town name:

'Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch'

8c
Write a script that takes user input and converts degrees Fahrenheit to degrees Celsius and prints the results to the screen (to 1 decimal place). The formula you require for the conversion is: T(°C) = (T(°F) - 32) × 5/9.

8d
Write a script that collects a value in pounds (£) and a percentage tax rate from the user and then writes to the screen the amount after tax is added. Since this is money, write out the results to two decimal places and include commas to separates thousands (000s). Name the script `tax_calculator.py`.

Exercise 9: user-defined functions

9a

Define a function called `fahr_2_cels` that converts degrees Fahrenheit to degrees Celsius. It should take a value in degrees Fahrenheit as input and return a Celsius value. The function should be called from the main body of a script named `temperature_calculatons.py`.

i) Write a documentation for the function `fahr_2_cels` to explain what it does.
ii) Assert that the input to the function is a `float,` else end the script with an appropriate message.

9b
In a similar fashion to the previous exercise, define a function called `calculate_tax` that takes a float value and returns the value after adding tax. Write documentation. Name the script `tax_calulations.py`. Also, if no tax value is supplied, set this value by default to 20%.

Exercise 10: methods

Using what you have learned so far, studying the cheat sheet and possibly online resources, complete the following exercises dealing with Python methods. Type your expressions into the Thonny command line.

10a
Write a string method to count the number of times “s” appears in the word “Mississippi”.

10b
Write a string method to capitalise the first letter of the text “much ado about nothing”.

10c
Write a string method to check whether all the characters in the text “O1OO1OO111OOO11O” are digits.

10d
Write a string method to check whether all the characters in the string “The Magic Roundabout” are lower case.

10e
Write **one-line** expression to convert “The Magic Roundabout” to lowercase and then check whether it is lower case.

10f
At what position does the word “razor” occur in the string “Stately, plump Buck Mulligan came from the stairhead, bearing a bowl of lather on which a mirror and a razor lay crossed.”

Exercise 11: tying it all together
Write a function that takes a string of nucleotides and reports the GC content of the string (as a fraction of 1). Save the script to a file called `gc_calculator.py`.

Extra points to consider:
Provide function documentation.
The function needs to handle upper/lowercase characters.

Chapter 4 – collections

Exercise 12: sets

Use the interactive window to:

12a
Create a set named dna containing the nucleotides: “adenine”, “cytosine”, “guanine”, “thymine”.

12b
Make a copy of the dna set and name it rna (use the copy() method to do this – you will find out later why you should make the copy this way!). Unfortunately, the bases are now not quite correct. Therefore, remove “thymine” and replace it with “uracil”.

12c
Using the appropriate set operators:

i) identify bases common to rna and dna
ii) identify bases that are not common to both sets

12d
i) Deduplicate the letters in the word ‘bookkeeper’
ii) Deduplicate the nucleotides: “AaaaaGgGGGCgcGG” (but treat lower/upper case as equivalent).

12e*
As briefly mentioned, there is a related datatype in Python known as a `frozenset`. Read the python documentation about this datatype. Now try to recreate the dna and rna data structures using a frozenset. What happens if we try to edit the rna frozenset? Is this what you expected?

(Hint: to create the frozenset, try passing a set object to the function `frozenset()`.)

Exercise 13: ranges

13a

Use the interactive window to:

i) Create a range of integers from 1 to 7 inclusive.
ii) Create a range from of integers -5 to 3 inclusive.
iii) Create a range of all the odd numbers from -3 to 5.
iv) Pass the range to a set from the previous exercise to view the actual numbers

Exercise 14: tuples

14a

Use the interactive window to:

i) Create a tuple containing the first six letters of the alphabet.
ii) Create a tuple of the integers from 0 to 1000 inclusive (there is a quick way to do this!)
iii) Run the built-in function `max()` on your tuple to retrieve the maximum value
iv) Create a tuple that contains the following letters the letters ‘A’, ‘B’, ‘C’, repeated 10 times.
v) Create a tuple containing only your first name
vi) Check that you have created a tuple in iv)
vii) Tuples have 2 methods: `count()` and `index()`. Write code to work out what these methods do. Use the following tuple to help you:
sample = (1,1,2,1,1,2,2)

Exercise 15 – lists

Use the interactive window to:

i) Create a list of the planets, in order of distance from the sun:
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
ii) Check you have indeed created a list in i).
iii) Using the list, print out the name of the planet closest to the sun.
iv) Print out the number of planets.
v) Print out the name of planet furthest from the sun.
vi) Find the position of Jupiter.
vii) Print out the names of the planets after Jupiter.
viii) Check whether Pluto is included in this list.
ix) Let’s add Pluto as the planet furthest from the sun.
x) Print out the name of every third planet from the sun.
xi) Print out the names of the planets in alphabetical order, but do **not** change the order of the list. (Hint: compare the string sort() method to the sorted() function. Also, what does the string sort() method return?)

xii) Print out the names of the planets from the furthest from the sun to the closest.
xiii) Copy the list to another list called `planets_copy`.
xiv) Remove the last planet from the list `planets`.
xv) Change the order of planets to be arranged alphabetically.
xvi)

Exercise 16 - dictionaries

Use the interactive window to:

i) Create a dictionary called `group1_elements` of the Group I chemical elements:
H: Hydrogen
Li: Lithium
Na: Sodium
K: Potassium
Rb: Rubidium
Cs: Caesium
F: Francium
ii) Retrieve from the dictionary the name of the chemical element with symbol Na.
iii) Check whether the chemical element U is in our list.
iv) Print out the keys of our dictionary.
v) Make a copy of `group1_elements` called `alkali_metals`
vi) Remove H from alkali metals
vii) Let's empty the dictionary alkali_metals.

Exercise 17 – files

i) Write a script called `read_file.py` that opens the file `lonely_cloud.txt` and writes the file contents to the screen. (Hint, the will use the `readlines()` method.)

ii) Play around with the code you have just created and work out what datatype the `readlines()` method has returned.

iii) Using this new-found knowledge, print **every other line** from the poem to the screen in a script called `read_file2.py`.

iv) Edit read_file2.py so that in addition to printing to the screen the edited poem is written to a file called `wordsless.txt`. (Hint, this will use the `writelines()` method.)

v) There is also a method name `read()`. Does this return a different datatype than `readlines()`?

Exercise 18 – lambda function

Write a lambda function to sort the names: 'William', 'Catherine', 'Harry' and 'Meghan' by their last letters. Save the script as `royals.py`.

Chapter 5 – conditionals, loops and iterations

Exercise 19 – conditionals

19a
Create a script called `conditional_tester.py` and declare a name called `my_value` with a numerical value of your choosing. Then perform the **one-alternative conditional tests** listed below. Print an appropriate message to the screen if the conditional test returns a `True` result.

i) Write a conditional expression that checks whether `my_value` is equal to 10
ii) Write a conditional expression that checks whether `my_value` is less than 5.
iii) Write a conditional expression that checks whether `my_value` is greater than or equal to 1001.
iv) Write a conditional expression that checks whether `my_value` is not equal to 7.

19b
Create a new script named `password.py` that prompts the user to enter a password. The correct password should be 'Joshua'. An incorrect password returns the response 'Access denied', whereas a correct response returns 'Greetings Professor Falken!'

19c
The grade boundaries for an exam were:
70% or more : A
60 – 69% : B
50 – 59% : C
40 – 49 % : D
Less than 40: Unclassified

Write a script called `exam.py` that uses a multi-test conditional to classify a score using the above schema.

19d*
It is common knowledge that a leap year is a year exactly divisible by 4. However, this is not the whole story, for if a year is divisible by 100 then it is **not** a leap year. However, even that is not the whole story, for if a year is divisible by 400 then it **is** a leap year once more. Write a Python function to decide whether a year is a leap year.

Provide documentation for the function and assert that the input to the function is an integer, else the function should terminate with a relevant message. Write the code in a script called `leap_year.py`.

Exercise 20 – while loops

20a
Write a script called `integer_sum.py` that calculates the sum of the first 100 integers (i.e. 1 to 100) using a `while` loop. This total should be printed to the screen.

20b
Modify the script `password.py` so the script keeps asking for a password if a wrong password is entered by the user. If the correct password is entered, the greeting message should be displayed and the `while` loop is broken. Call this new script `password2.py`.

20c
Write a script named `colours.py` which opens the file `colours.txt` and uses a `while` loop and a `set` to deduplicate the list of colours (retaining one representative copy) in the file. Print the list of unique colours to the screen.

Exercise 21 - iterations

21a
Modify the `colours.py` script so it iterates over the file with a `for` loop. Also, print out the unique list of colours using a `for` loop. Call this new script `colours2.py`.

21b
Modify `colours2.py` so that the script now counts how many times each colour appears in the file (hint: use a `dictionary` with the `get` method). Also, use `enumerate` to keep track of the total number of colours in the file. Print this total to the screen, and also report the tally of each colour. Call this new script `colours3.py`.

21c
A chessboard comprises 64 squares, with 8 columns (A-H) and 8 rows (1-8). Using nested iterations, print out the chessboard references to the screen:

```
a8 b8 c8 d8 e8 f8 g8 h8
a7 b7 c7 d7 e7 f7 g7 h7
a6 b6 c6 d6 e6 f6 g6 h6
a5 b5 c5 d5 e5 f5 g5 h5
a4 b4 c4 d4 e4 f4 g4 h4
a3 b3 c3 d3 e3 f3 g3 h3
a2 b2 c2 d2 e2 f2 g2 h2
a1 b1 c1 d1 e1 f1 g1 h1
```

Iterate over a `range` to generate the numbers. Call this script `chessboard.py`.

21d*
Let's continue with the chess theme: suppose there is an empty chessboard, save for a solitary king in the bottom left corner. We then give his majesty a series of instructions to move one space at a time – either forwards (F), backwards (B), right (R) or left (L). We provide the instructions in a string: `'FFFBFFRRRLRBFFBBBBRRLRLRLRFBBFBFBLRLBBFR'`.

Will the king, **at any point**, leave the board as a result of those instructions? If so, from which side of the board will the king leave? Write a Python script named `king.py`, that **iterates over the string**, to answer this question.