Collaborative Document

2025-06-03 Introduction to Python

Welcome to The Workshop Collaborative Document.

This Document is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

This is the Document for today: https://edu.nl/k8axf

Collaborative Document day 1: https://edu.nl/k8axf

Collaborative Document day 2: https://edu.nl/vehjx

Code of Conduct

Participants are expected to follow these guidelines:

- Use welcoming and inclusive language.
- Be respectful of different viewpoints and experiences.
- Gracefully accept constructive criticism.
- Focus on what is best for the community.
- Show courtesy and respect towards other community members.

If you feel that the code of conduct is breached, please talk to one of the instructors (if the complaint is for one of the participants) or send an email to training@esciencecenter.nl (if the complaint is for one of the instructors).

Certificate of attendance

If you attend the full workshop you can request a certificate of attendance by emailing to training@esciencecenter.nl.
Please request your certificate within 8 months after the workshop, as we will delete all personal identifyable information after this period.

⚖ License

Getting help

To ask a question, raise your hand in zoom. Click on the icon labeled "Reactions" in the toolbar on the bottom center of your screen,
then click the button 'Raise Hand '. For urgent questions, just unmute and speak up!

You can also ask questions or type 'I need help' in the chat window and helpers will try to help you.
Please note it is not necessary to monitor the chat - the helpers will make sure that relevant

questions are addressed in a plenary way.
(By the way, off-topic questions will still be answered in the chat)

Workshop website

link (https://esciencecenter-digital-skills.github.io/2025-06-03-dc-socsci-python-odissei/)

Setup

link (https://datacarpentry.github.io/python-socialsci/index.html#setup)

Download files

link (url)

Instructors

Helpers

□□□□□□□□□□♂□ Roll Call

Icebreaker: your favourite dish

Pizzoccheri (https://en.wikipedia.org/wiki/Pizzoccheri)

https://vidarbergum.com/recipe/imam-bayildi-turkish-stuffed-aubergines/

□ Agenda

Time	Topic
09:00	Welcome and icebreaker
09:15	Introduction to Python and Python basics
10:20	Coffee break
10:35	Python control structures
11:15	Coffee break
11:30	Creating reusable code
12:45	Wrap-up
13:00	END

Exercises

Exercise: Arithmetic and printing

Create a new cell and paste the following code into it:

```
print("a =", a, "and b =", b)
print(a + 2*b)
print(a + (2*b))
print((a + b)*2)
```

1. Remove all of the calls to the print function so you only have the expressions that were to be printed and run the code. What is returned?
2. Now remove all but the first line (with the 4 items in it) and run the cell again. How does this output differ from when we used the print function?

Bonus if you are done early:

- Practice assigning values to variables using as many different operators as you can think of. (see [this list of arithmetic operators (https://www.w3schools.com/python/gloss_python_arithmetic_operators.asp)](https://www.w3schools.com/python/gloss_python_arithmetic_operators.asp))
- Create some expressions to be evaluated using parentheses to enforce the order of mathematical operations that you require

Exercise: Lists and ranges

1: List indexing

1. Create a cell with the code below.

```
num_list = [4,5,6,11]
```

2. Select the 1st element from this list. (Your code should return 4)
3. Select the last element from this list. (Your code should return 11)
4. Make a new list with the second and fourth element in this list. (Your code should return `[5,11]`)
5. (Bonus): Use the 'insert()' method to insert `5.5` in between 5 and 6.
6. (Bonus): Which method removes an element at a specified position?
7. (Bonus): Sort the list in descending order.

Exercise: if-statements

1. Start with the following code:

```
apple_cost = 0.5
bread_cost = 2.5
money = 2
```

2. Write an `if`-statement, replacing the `____` in the code below, that checks whether you can buy a bread with your money:

```
if _________
    print("I can buy bread!")
```

3. Add an `elif` to your statement, where you check if you can buy an apple instead.

```
elif _________
    print("At least I can buy an apple!")
```

4. Bonus: write a final `else` for the tragic scenario where you can buy neither bread nor apple.

Exercise: for-loop

Suppose that we have a string containing a set of 4 different types of values separated by , like this:

```
variablelist = "01/01/2010,34.5,Yellow,True"
```

Research the `split()` method and write a for-loop that prints each of the 4 components of `variablelist`

Exercise: Creating functions

1. Write a function definition to calculate the volume of a cuboid. The function will use three parameters h, w and l and return the volume.
2. Supposing that in addition to the volume I also wanted to calculate the surface area and the total combined length of all of the edges. Would I (or should I) have three separate functions or could I write a single function to provide all three values together?

Collaborative Notes

Working with notebooks

There are different types of cells. The 2 main types we will look at today is Python and Markdown. Markdown allows you to write text and headers. In the Python code blocks, you can type some code.

In Python, types are assigned when you create them. You can look at the type by:

```
a = 2
b = 3.14
type(b)
```

You can print by doing something like this:

```
print("a = ", a, "and b = ", b)
```

This will print this sentence: `"a = 3 and b = 3.14"`
The print function serves to print out the values of variables that have already been assigned. (So Python needs to know that it exists).

Python has all of the standard operators, such as: `+, -, *, /`
For the power of: **

You can get help within Jupyter with the help function:

```
help(print)
```

You can also start typing the function name and then do: `shift + tab`

You can change the type of your variable after it has been assigned:

```
float(a) # converts int to float
int(b) # converts float to int
int("4") # converts string to int
```

About strings

```
mystring = "Hello, World!"

len(mystring) # Finds length of the string
mystring.upper() # Converts string to upper case (you can also use .lower())
mystring.find("orl") # Finds where those characters are
mystring.split(",") # Splits the string into separate parts where "," is
located
```

You can add strings together to concatonate them:

```
"A" + "B" + "C"
```

Booleans

These are `True` and `False`
Most of the time we use booleans as a result of analysing things.

```
'hello' == 'HELLO'
```

is `False`
but:

```
'hello' == "hello"
```

is `True` as both `` and "" can be used interchangeably in Python

There are several operators to check equivalence:

```
b is None #Checks if b is equal to None
3 != 77 #Checks if 3 is not equal to 77
3 < 1 #Checks if 1 is larger than 3
```

Lists

```
list1 = [6,54,89]
list2 = [5,3.14,"hello", b] # Contains diff types
```

You can check the length of a list and also append lists together

```
len(list2)
list1 + list2
```

You can also only take one element from the list:

```
list1[0] # Get first item
list1[:-1] # Get last item
```

Python is zero based, so `list[0]` gives you the first element and `list[1]` gives you the second, etc.

You can generate a range of numbers with:

```
r = range(5)
```

If we generate a list from that:

```
list(r)
```

We will see that the list is `[0,1,2,3,4]`
You can also get a range from a list:

```
l = list(range(20)) # Gives you a list with 20 items from 0-19
l[5:10] # Gives you the list from elements 5 to 10
```

Dictionaries

These are key, value pairs:

```
d = {a: "Hello", b:"World"}
```

Python control structures

if statements

This is when you want to execute some code based on the value of the previous statement. The syntax is as follows:

```
if expression:
    statement 1
    ...
    statement 2
xxxx # outside the if block
```

In Python, the indentation defines what is within and outside of the if statement. Python is flexible in how many spaces you use but you should be consistent. Best practice is to use 4 spaces (tabs can be interpreted differently across systems)

```
x = 5
threshold = 4
if x > threshold:
    print(x, "is larger than", threshold)
```

```
high_threshold = 6
if x > high_threshold:
    print(x, "is also larger than", threshold)
```

```
mid_threshold = 5
if x == mid_threshold:
    print(x, "is equal to", threshold)
```

You can also include an else statement to execute a different bit of code depending on the condition:

```
if x < threshold:
    print("below threshold")
else:
    print("above or equal to threshold")
```

Or you can add multiple conditions:

```
if x < threshold:
    print("below threshold")
elif x == threshold:
    print("equal to threshold")
else:
    print("above threshold")
```

Order here matters, as the conditions will be executed in order.

While loops

This will run a loop until a condition is met. The syntax is as follows:

```
while condition:
    statements
    ...
```

```
n = 10
i = 1
cur_sum = 0

while i <= n:
    cur_sum = cur_sum + i
    i = i + 1
print(cur_sum, "is the sum of all integers from 1 to", n)
```

Make sure that you allow the loop to break at some point of it will run forever.

for loops

Similar to while loops but instead you loop over a sequence (such as a list). The syntax is as follows:

```
for variable in sequence:
    statements
    ....
```

```
for i in [1,2,3]:
    print(i)
```

```
for i in range(10):
    print(i)
```

```
for i in range(4,10):
    print(i)
```

You can use the step argument in range. In this example we go in steps of 2:

```
for i in range(4,10,2):
    print(i)
```

You can also loop over strings:

```
string = "the quick brown fox"
for i in string:
    print(i)
```

Which will print out every letter. We can istead split the string into words and it will iterate over words instead:

```
string = "the quick brown fox"
for i in string.split(" "):
    print(i)
```

Reusable code

We're going to write a function to get the number of items in a string.

```
def get_item_count(items: str, sep: str) -> int:
    """Splits 'items' using separator 'sep' and returns the number of elements.
    """
    item_list = items.split()
    num_items = len(item_list)
    return num_items
```

`items: str` the `str` indicator here is called a type annotation, it says in which format the input parameter should be included. It is optional and can be excluded, but it is helpful for anyone using your code.

The text in between """ """ is called a docstring, these are a way to document your function, by giving the input parameters and what the function should return. The triple quotes are for multistring delimiters in Python.
There are several standard formats for it, be consistent. This is also optional but is a good habit to have.

Let's now run the function:

```
items_owned = "bicycle, car, unicycle"
get_item_count(items_owned)
```

Installling Libraries

To install a package, from terminal write:

```
pip install [package-name]
```

For example:

```
pip install pandas
```

To import libraries in Python:

```
import [package-name] # imports the whole library
import pandas as pd # imports library with an alias
from math import sqrt # imports only certain functions
from matplotlib import pyplot as plt # imports a function and gives it an alias
```

You can use * to import everything but it is not recommended.

Classes

Classes are a way to collect functions (you might see the term method used, this is a function within a class).

```
class Cuboid:
    """Describes a cuboid volume of a certain heigh, width and length"""
    def __init__(self, h, w, l):
        self.height = h
        self.width = w
        self.length = l

    def volume(self):
        """Returns volume of a cuboid"""
        return self.height * self.width * self.length

    def surface_area(self):
        """Returns the surface area of a cuboid"""
        return 2 * (self.height * self.width +
                    self.width * self.length +
                    self.height * self.length)
```

The `__init__` defines what the object is. In this case we are defining a cuboid that has a height, width and length. The following methods will then be defined to manipulate that cuboid (or object).

The key word 'self' is used as the first argument for every method in a class.

To then use this class we can do:

```
cuboid = Cuboid(3,4,5)
c.volume()
c.surface_area()
```

Feedback

Resources

- JupyterLite (https://esciencecenter-digital-skills.github.io/socsci-python-jupyterlite/)
- Python documentation (https://www.python.org/)
- PyPi (Python packages) (https://pypi.org/)
- Workshop material (https://datacarpentry.github.io/python-socialsci/)