



BSc Intro 2022 Python Tutorial

Wael Alkakhi, Ishan Pokharel, Chris Scheulen, Sreelakshmi Sindhu

II. Physikalisches Institut, Georg-August-Universität Göttingen

2022-03-09

- The interpreter: A simple way of testing things out
- Variables in Python
- Indentation or how I learned to stop worrying and love C++'s Semicolons
- Control flow: The ifs and loops of programming
- Using functions and classes in Python
- Multiple files and imports

Now it's your turn! We've prepared some exercises for you in the following. Try them out!

If anything is unclear or if you are stuck: Ask us.

Maybe also try using `git` to version control your code. That will also allow us to look at it together if you are stuck somewhere!

- On lxplus (or most other clusters), you are presented with a certain selection of Python modules
 - Load as follows (on lxplus):
`$ setupATLAS; lsetup "python <your version>"`
 - Look-up versions as discussed yesterday:
`$ showVersions python`
 - In general: OK, but often it would be nice to install some custom modules (e.g. tensorflow for Machine Learning), choose different versions (e.g. for numpy), or have multiple versions of Python for different purposes
- ⇒ Python virtual environments are your friend!
- For setting them up:
 - Create venv: `$ python3 -m venv <path/to/venv>`
 - Activate venv: `$ source <path/to/venv>/bin/activate`
Notice how (<venvname>) should now be in front of your shell prompts!
 - Add modules: `$ pip install {-U} <module> (-U for updates)`
 - Try it out! Build yourself a Python3 venv with the following modules (for tomorrow/Friday!):
ruamel.yaml, matplotlib, uproot, pandas, tensorflow, scikit-learn, tables, pydot
- But:** Start by updating pip: `$ pip install -U pip`

1. Open the python interpreter and print out 'Hello World!'
2. Now, do the same thing from a `macro.py` file
3. Probably, you needed to use the following for the last task:

```
$ python macro.py
```

There is also another way involving a so-called *shebang*. Try it out with that!

4. Use your file to print out the integers 1 to 20
5. Now only print the even integers from 1 to 20
6. Can you store these integers in a file?

Hints:

- Python has a concept similar to the main-method in C++. Try the following for that:

```
if __name__ == "__main__":
```

- To make a file executable with Linux, you also have to change it's permissions. For that you can use `chmod +x <executable>`
- Try out what the following command does in the interpreter:

```
range(20)
```

1. Write your name, age, and level of study in a dictionary
2. Ask the other people in the room about their corresponding info and compile a list of all people
3. Print the first names of all people in the room separated by commas
4. Save the information of everybody in the room in a text file

Hints:

- The structure of a dictionary is {<key>: <value>, ...}, that of a list is [<value>, ...]
- You can open files using one of the following:
 using `open(<filename>, <access_option>)` as `f`:
 <STUFF>
(preferred, since the file will be closed correctly if an error happens!)
 `f = open(<filename>, <access_option>); <STUFF>; close f`
- For reading/writing, you can use `f.read()`, `f.readlines()`, `f.write(<text>)`, and `f.writeline(<text>)`. Try out what the different versions do in an editor!

1. Write a method with which you can add more people to the room list
2. Make a class `person` and `personlist`, which have the same contents as the previous dictionary and list
3. Add the `add-person` method to the `personlist` class
4. Also add a `__str__`-method to the `person` class, with which you can print the information contained in the class

Hints:

- To create methods and classes, you need to use `def <method>(<arg>, ...):` and `class <Class>:`
- Classes can be initialized with the magic method `def __init__(self, <arg>, ...)`
- If you assign `self.<name> = <value>`, the variable you assign can be used throughout the class
- `__str__` is another magic method without args (except `self!`), which should return a string and is automatically called by `print(<Class>)`

1. Put the classes you just created in their own files and the main-method in another one
2. Use `sys.argv` to add people to the `personlist` class from the commandline
3. Check out python's `argparse` standard library and write some parameters for your executable
4. Add methods to load and save your `personlist` as `json` and `yaml` files

Hints:

- You can make python files and libraries accessible to each other using `import`-statements
- Try out what `sys.argv` does with a simple macro if you are unsure
- The references for [argparse](#), [json](#), and [yaml](#) can tell you how each of these packages works!

If you still have time left, try out one of these tasks:

- Use `numpy`-arrays to check out the anticommutators of the γ -matrices. Do they behave as expected?
- Build a class 'Uncertainty' which allows adding in quadrature
- Expand your person-class to also include a subclass 'Student' with inheritance. At this point, the argument 'level-of-study' of course only makes sense for students.

- Overall, Python is a very user-friendly programming language
- Now used extensively for Data Science → Connection with C++ based ROOT via uproot module
↳ More about that tomorrow
- Of course we could only give you a very brief overview of Python. If you are stuck during your thesis, just look online for help or ask your friendly PhD student!

Thanks for your attention