Introduction to Programming in Python

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Preliminaries

General Information

- My name is Simon Broda. Email: simon.broda@hslu.ch.
- Format of this course: 14 lectures of 2h each, mix between theory and practice.
- Final grade based on a group assignment (groups of two; 50%) and a final exam (open book, 90min; 50%).
- Additional exercises will be made available but not graded.

Material

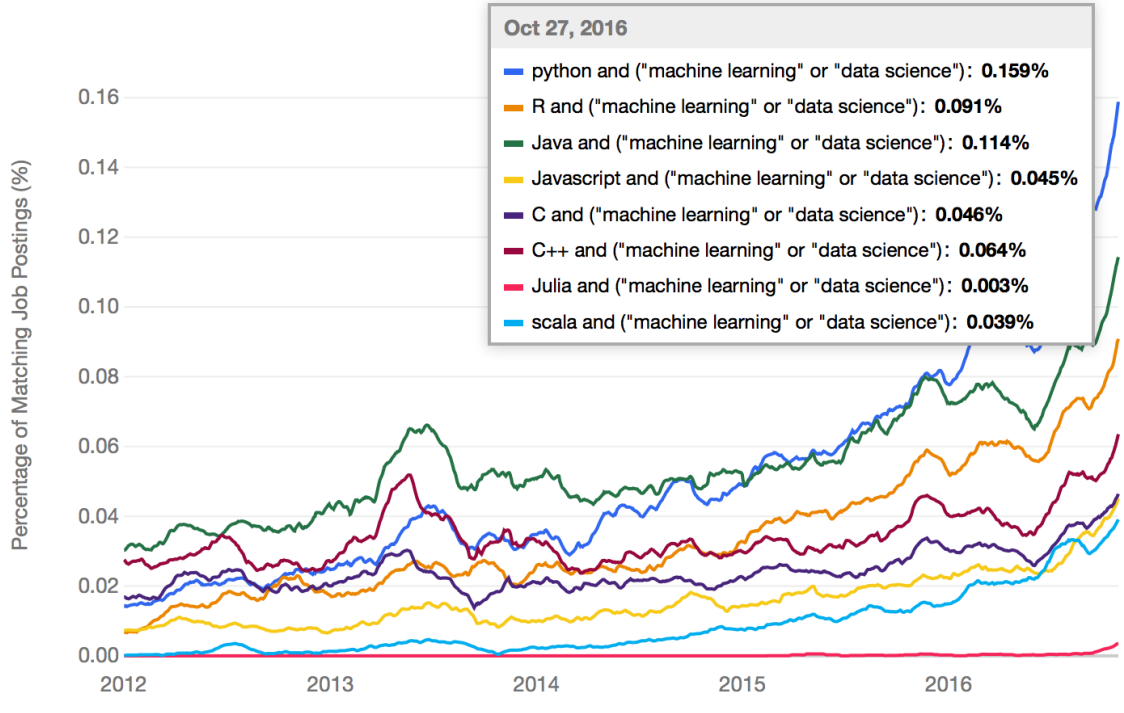
- These lecture slides. Available on Github.
- Website: https://python-course.eu/
- Sources for additional exercises:
    - https://holypython.com/beginner-python-exercises/
    - https://pythonbasics.org/exercises/
- Further reading:
    - Python documentation

Introduction to Python

Why Python?

- General purpose programming language, unlike, e.g., Matlab®.
- High-level language with a simple syntax, interactive (*REPL*: read-eval-print loop). Hence ideal for rapid development.
- Vast array of libraries available, including for scientific computing and finance.
- Native Python is usually slower than compiled languages like C++. Alleviated by highly optimized libraries, e.g. NumPy for calculations with arrays.
- Free and open source software. Cross-platform.
- Python skills are a marketable asset: most popular language for data science.

Job Postings on Indeed.com



Source

But Python can do all kinds of things...

In [ ]:

```
# uncomment the next line if you don't have googlesearch installed yet
# !conda install -c conda-forge -y googlesearch
from googlesearch import search
query = "best course for python"
for i in search(query, tld="com", num=10, stop=10, pause=2):
    print(i)
```

In [ ]:

```
# !pip install instaloader
import instaloader
from IPython.display import Image
from os import walk
d = instaloader.Instaloader()
profile_name = 'loredana'
d.download_profile(profile_name, profile_pic_only = True)
filename = next(walk("./" + profile_name), (None, None, []))[2][0]
pil_img = Image(filename='./' + profile_name + '/' + filename)
display(pil_img)
```

In [ ]:

```
# !conda install -y pytube
# !pip install moviepy
from pytube import YouTube
from IPython.display import Audio
import moviepy.editor as mp
url = "https://www.youtube.com/watch?v=gdsUKphmB3Y"
yt = YouTube(url)
ys = yt.streams.get_highest_resolution()
a = ys.download("./")
clip = mp.VideoFileClip(a)
clip.audio.write_audiofile('out.mp3')
Audio("out.mp3", autoplay=True)
```

These examples are taken from https://dev.to/unitybuddy/9-amazing-things-to-do-with-python-1ln5. Check it out, there are more!

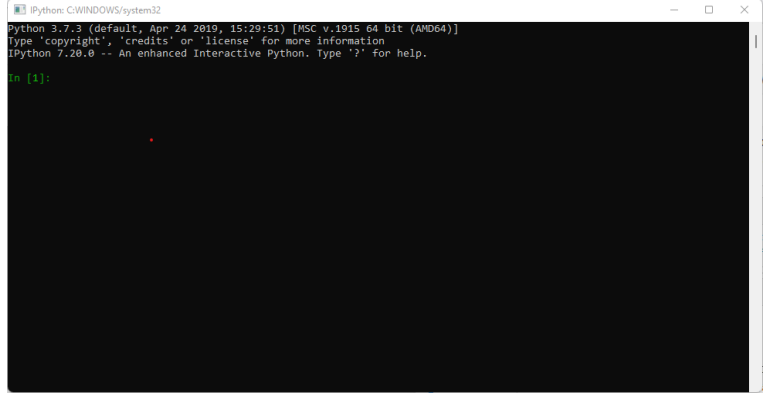
Obtaining Python

- Anaconda is a Python distribution, developed by Continuum Analytics, and specifically designed for scientific computing.
- Comes with its own package manager (conda). Many important packages (the *SciPy stack*) are pre-installed.
- We will install it together right now. You can find it here. I recommend adding it to your `PATH` upon installation.
- Optional: Install the RISE plugin to allow viewing notebooks as slide shows:

In [ ]:
```
# uncomment the next line to install. Note: "!" executes shell commands.
# !conda install -c conda-forge rise
```

IPython Shell

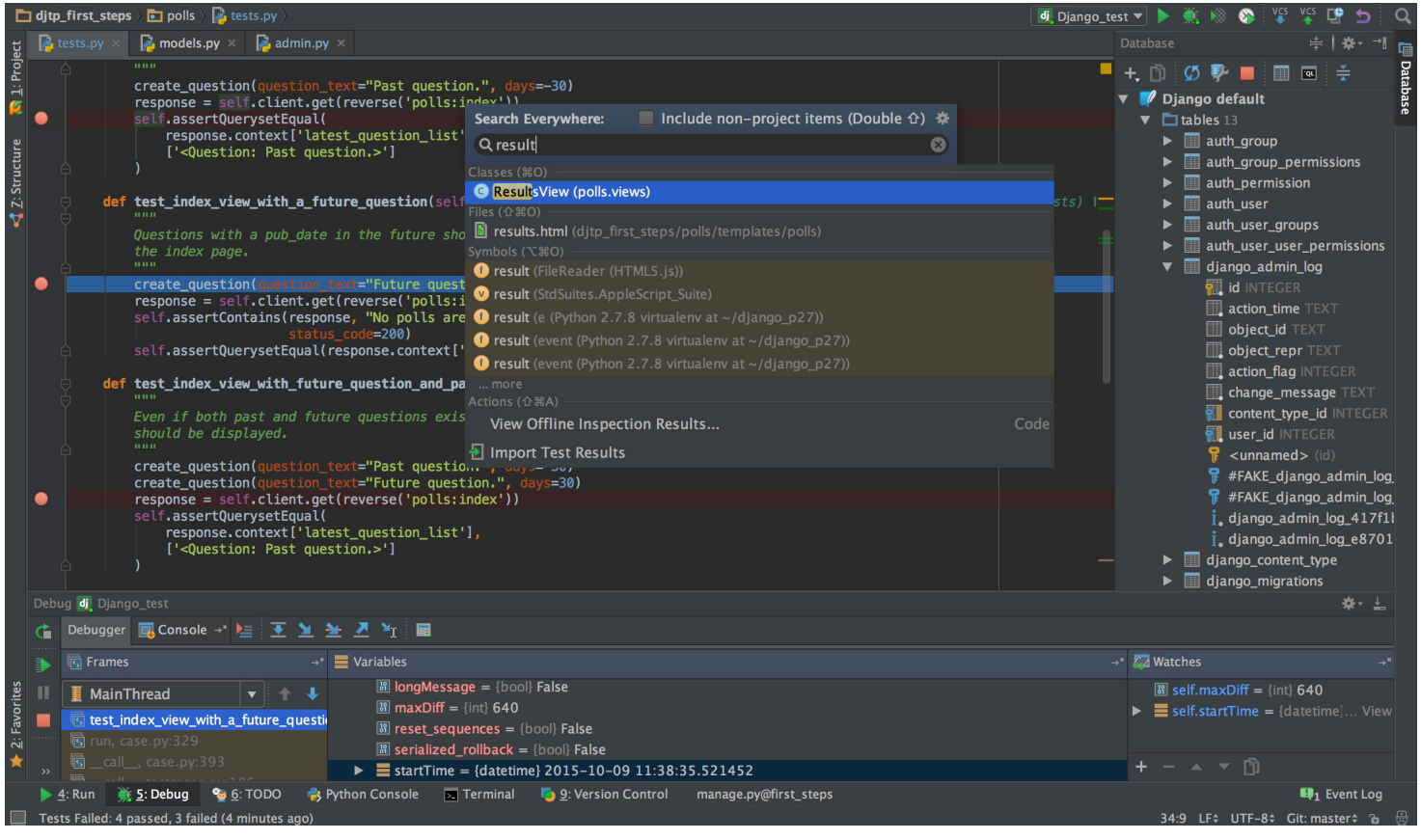
- Python features a *read-eval-print loop* (REPL) which allows you to interact with it.
- The most bare-bones method of interactive use is via the *IPython shell*: You can start it by entering `ipython` on the command line (Windows; just enter `cmd` in the start menu search) or the terminal (MacOS; start it using Launchpad).



- For now, you can treat it as a fancy calculater. Try entering `2+2`. Use `quit()` or `exit()` to quit, `help()` for Python's interactive help.

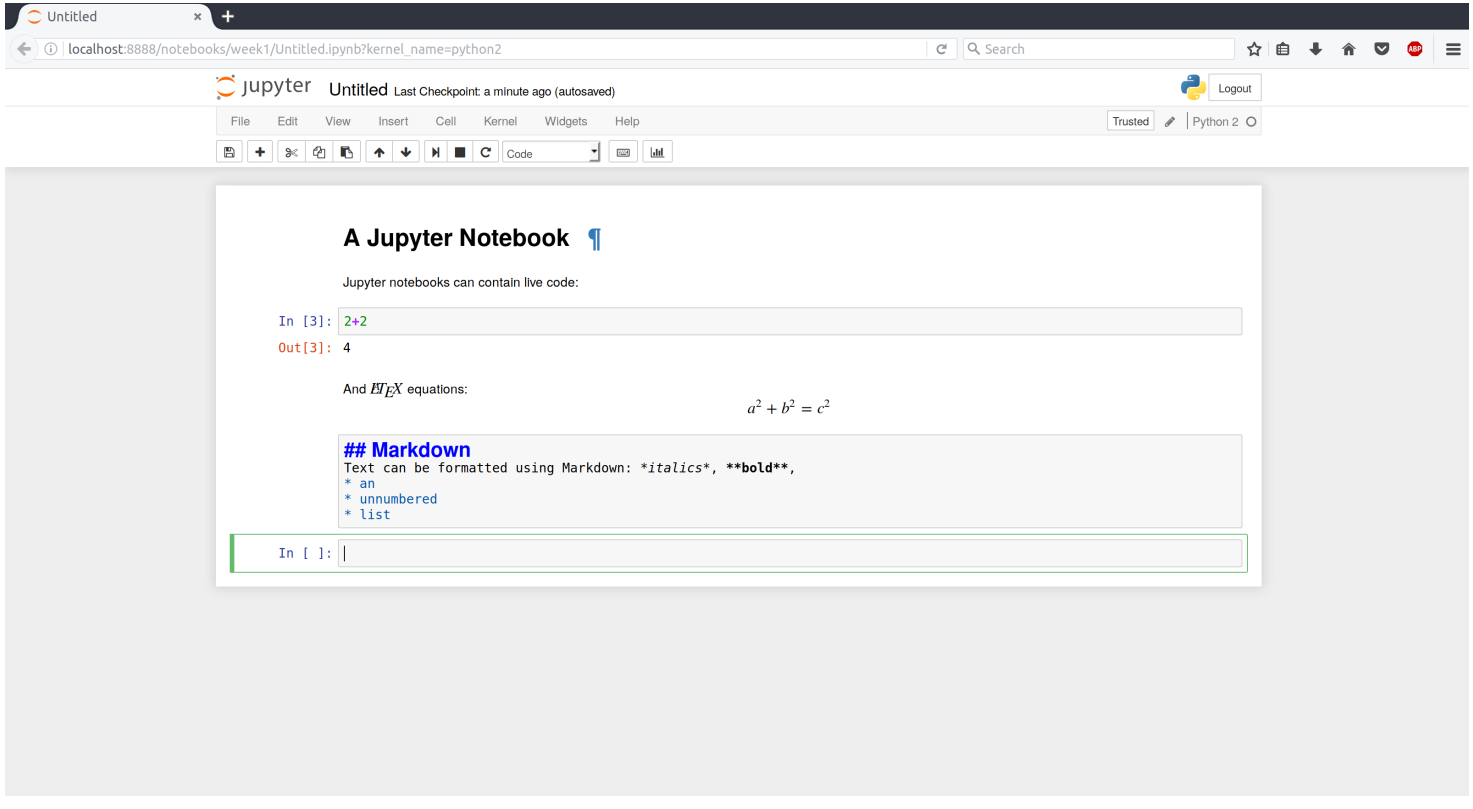
Writing Python Programs

- Apart from using it interactively, we can also write Python *programs* so we can rerun the code later.
- A Python program (called a *script* or a *module*) is just a text file, typically with the file extension `.py`.
- It contains Python commands and comments (introduced by the `#` character)
- To execute a program, do `run filename.py` in IPython (you may need to navigate to the right directory by using the `cd` command).
- While it is possible to code Python using just the REPL and a text editor, many people prefer to use an *integrated development environment* (IDE).
- Anaconda comes with an IDE called *Spyder* (Scientific PYthon Development EnviRonment), which integrates an editor, an IPython shell, and other useful tools.
- An alternative is PyCharm, which we will be using later in this course.


djtp_first_steps
polls
tests.py
Django_test
tests.py
models.py
admin.py
Search Everywhere:
Include non-project items (Double ⇧)
result
Classes (⌘O)
ResultsView (polls.views)
Files (⇧⌘O)
results.html (djtp_first_steps/polls/templates/polls)
Symbols (⌥⌘O)
result (FileReader (HTML5.js))
result (StdSuites.AppleScript_Suite)
result (e (Python 2.7.8 virtualenv at ~/django_p27))
result (event (Python 2.7.8 virtualenv at ~/django_p27))
result (event (Python 2.7.8 virtualenv at ~/django_p27))
... more
Actions (⇧⌘A)
View Offline Inspection Results...
Code
Import Test Results
Database
Django default
tables 13
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
id INTEGER
action_time TEXT
object_id TEXT
object_repr TEXT
action_flag INTEGER
change_message TEXT
content_type_id INTEGER
user_id INTEGER
<unnamed> (id)
django_content_type
django_migrations
Debug Django_test
Debugger
Console
Frames
MainThread
Variables
longMessage = {bool} False
maxDiff = {int} 640
reset_sequences = {bool} False
serialized_rollback = {bool} False
startTime = {datetime} 2015-10-09 11:38:35.521452
Watches
self.maxDiff = {int} 640
self.startTime = {datetime}... View
4: Run
5: Debug
6: TODO
Python Console
Terminal
9: Version Control
manage.py@first_steps
Event Log
Tests Failed: 4 passed, 3 failed (4 minutes ago)
34:9
LF
UTF-8
Git: master

Jupyter Notebooks

- Another option is the *Jupyter notebook* (JUlia PYThon (e) R, formerly known as IPython notebook); this is what we will use in the coming weeks.
- It's a web app that allows you to create documents (`*.ipynb`) that contain text (formatted in Markdown), live code, and equations (formatted in $\LaTeX$).
- In fact these very slides are based on Jupyter notebooks. You can find them on my Github page.
- You can start Jupyter either from the Anaconda Navigator, or by typing `jupyter notebook` in the command line / terminal.


Untitled
localhost:8888/notebooks/week1/Untitled.ipynb?kernel_name=python2
Search
jupyter
Untitled
Last Checkpoint: a minute ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 2
Code
A Jupyter Notebook ¶
Jupyter notebooks can contain live code:
In [3]:
2+2
Out[3]:
4
And LaTeX equations:
$a^2 + b^2 = c^2$
Markdown
Text can be formatted using Markdown: *italics*, **bold**,
* an
* unnumbered
* list
In [ ]:

- A notebook consists of cells, each of which is either designated as Markdown (for text and equations), or as code.
- You should take a moment to familiarize yourself with the keyboard shortcuts. E.g., `enter` enters edit mode, `esc` enters command mode, `ctrl-enter` evaluates a cell, `shift-enter` evaluates a cell and selects the one below.
- Useful references:
    - Jupyter documentation;
    - Markdown cheat sheet;
    - Latex math cheat sheet.

Python Basics

Numbers

Math with integers works as you would expect:

In [ ]:
```
5 + 2
```

In [ ]:
```
2 - 4
```

In [ ]:
```
7 * (6 + 1) # brackets work as usual
```

In [ ]:
```
2 ** 3 # two to the third power
```

All in one cell:

In [ ]:
```
5 + 2
2 - 4
7 * (6 + 1)
2 ** 3
```

What happened? Jupyter will only print the result of the *last* expression in a cell. We can fix that by using the `print` function:

In [ ]:
```
print(5 + 2)
print(2 - 4)
print(7 * 7)
print(2 ** 3)
```

What about division?

In [ ]:
```
2 / 3
```

This works too, but it returns a different kind of number: a floating point number or `float`. This is true even when the division could in principle be done exactly:

In [ ]:
```
6 / 2
```

To a human, 3.0 (a `float`) and 3 (an integer or `int`) represent the same number, they are represented differently in memory; we say that these two objects have a different **type**. We can find the type of an object like this:

In [ ]:
```
type(3)
```

In [ ]:
```
type(3.0)
```

Math with floats can be a bit tricky, because they are represented with finite precision, which means that not all numbers are representable:

In [ ]:
```
1 - 0.9
```

Variables

- A variable is a named memory location. It is assigned using " `=` "

In [ ]:
```
a = 2
b = 4
c = a + b
print(c)
```

Easy enough. Can you guess what the following does?

In [ ]:
```
a = 2
a = a + 1
print(a)
```

In [ ]:
```
a = 2
a += 1  # shorthand for a = a + 1
print(a)
```

- Variable names can be made up from letters, numbers, and the underscore. They may not start with a number. Python is case-sensitive: `A` is not the same as `a`.

Assignment versus equality

We just saw that variables are assigned using `=`.

In [ ]:
```
a = 3
print(a)
```

What if we want to compare if two numbers are equal? First attempt:

In [ ]:
```
# uncomment the next line and run the cell
# 3 = 3
```

This obviously didn't work. The correct way is to use `==`:

In [ ]:
```
3 == 3
```

In [ ]:
```
1 == 2
```

The returned object is of type `bool` (a "Boolean")

In [ ]:
```
type(True)
```

- A `bool` can take one of two values: `True` or `False`.
- They are returned by *relational operators*: `<`, `<=`, `>`, `>=`, `==` (equality), `!=` (inequality), and can be combined using the *logical operators* `and`, `or`, and `not`.

In [ ]:
```
1 <= 2 < 4
```

In [ ]:
```
1 < 2 and 2 < 1
```

In [ ]:
```
not(1 < 2)
```

Strings

- Strings hold text. They are constructed using either single or double quotes:

In [ ]:
```
s1 = "Python"
s2 = ' is easy.'
s3 = s1 + s2  # Concatenation
```

In [ ]:
```
type(s3)
```

This doesn't work:

In [ ]:
```
a = 3 # an int
b = "4" # a string
# uncomment and run:
# a  + b
```

We have to convert the string first:

In [ ]:
```
a + int(b)
```

We can also convert the other way:

In [ ]:
```
a = 3
b = str(a)
```

In [ ]:
```
type(b)
```

This is useful for printing:

In [ ]:
```
height = 1.89
print("I am " + str(height) + "m tall.")
```

One way to obtain a string is to ask the user for input:

In [ ]:

```
mystr = input("What's your name? ")
print(mystr)
```

Exercise

Write some code in the cell below that asks the user for their age, and then prints the age in dog years (i.e., divided by 7).

Example input:

```
What's your age?
```

If the user enters `28`, then this should result in the following output:

```
Your age in dog years is 4.0.
```

Note that `input` always returns a string, so you have to convert it to `int` (or `float`) to do math with it.

In [ ]:

Homework

Exercises 1-4 from https://holypython.com/beginner-python-exercises/