

Preparation for Class

- Class repo is at <https://git.ee.ethz.ch/python-for-engineers/class-fs20>.
- If anybody has NOT yet succeeded in forking the class repo and adding us as project member (minimum Reporter access level) please come talk to me or any of the assistants before the class!
- Pull in changes from the class repo to get this set of slides and today's theory and exercise notebooks:
 - execute the following statement in your Git folder:
\$ cd ~/class-fs20
\$ git pull upstream master
then type :wq followed by Enter to finish merging
 - we start when everyone has succeeded in pulling today's materials



Python for Engineers

- get productive in the classroom, in the lab and at home

03.03.2020 – Python Basics 1

Raphael Schwanninger, raphael.schwanninger@ief.ee.ethz.ch

Content

- Repetition from Lecture 1
- Introduction to Python Programming
 - Simple and combined datatypes
 - Control flow
 - Functions
- Exercise time!

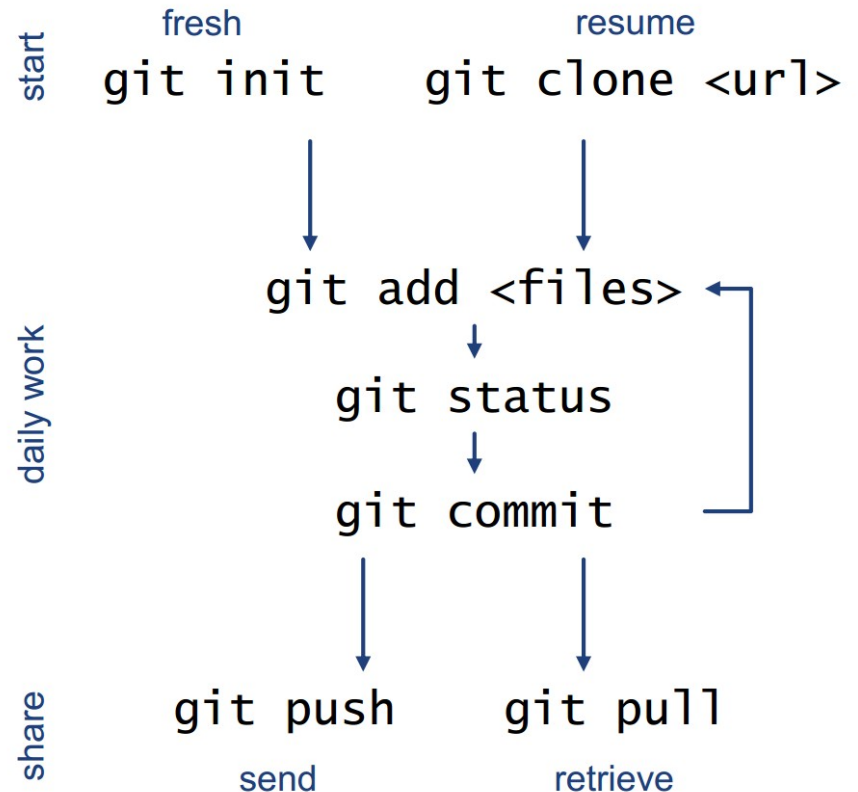
Repetition: Linux command-line

- Refer to the command manual whenever in doubt, or to the cheatsheet provided in Lecture 1.
 - `man <command-name>`
- Frequent commands:
 - `ls, pwd, mkdir, rm, top, grep, kill, ssh, cp, mv`
- Tab: autocomplete
- Ctrl-C: stop current command
- Highlight with mouse & middle mouse button: copy & paste

Repetition: Git

- Git is a hierarchical, decentralized source code management system.
- Keep track of changes to human-readable files
 - source code, scripts, TeX
- Keep old versions of each file with possibility to restore.
- Collaborate with others on a large scale
 - Git was devised by Linus Torvalds to coordinate work on the Linux kernel with thousands of contributors
- Fine-grained control over conflict resolutions
 - Happens all the time if you collaborate with others

Git operation in the Linux console:



Wrap-up: make your life easier

Exercise (5 min)

- Most of you have their forked projects on the GitLab <https://git.ee.ethz.ch> set to “internal” visibility. You might want to change this to “private”.
 - look under your project → Settings → General → Permissions

The screenshot shows the GitLab project settings interface. On the left is a sidebar with navigation links: Merge Requests (0), Wiki, Snippets, Settings (selected), General (selected), Members, Integrations, and Repository. The main content area is titled 'Project avatar' with a circular avatar containing the letter 'C'. Below the avatar is a 'Choose file...' button and the text 'No file chosen' and 'The maximum file size allowed is 200KB.' A 'Save changes' button is below that. The next section is 'Visibility, project features, permissions' with a 'Collapse' button. It contains a description: 'Choose visibility level, enable/disable project features (issues, repository, wiki, snippets) and set permissions.' Below this is a 'Project visibility' section with a dropdown menu currently set to 'Internal'. A black arrow points from the text 'Set to private' at the bottom right of the slide to this dropdown menu. Below the visibility section is an 'Issues' section with the text 'Lightweight issue tracking system for this project'.

Set to private

Wrap-up: make your life easier

Exercise (5 min)

- Modify your terminal config file to define an alias:
\$ nano ~/.bashrc (open the config file with the nano editor)
Then add these three lines to the end of the file:
alias anaconda='source /usr/local/anaconda3/bin/anaconda3_env.sh'
export EDITOR="nano"
export VISUAL="nano"
Use Ctrl+O and Ctrl+X to save and quit nano.
- Re-open your terminal to activate the changes.
- You can now activate the Python environment with the command
\$ anaconda
- Git and other command-line tools now use the nano editor instead of vim.

Anatomy of a piece of Python code:

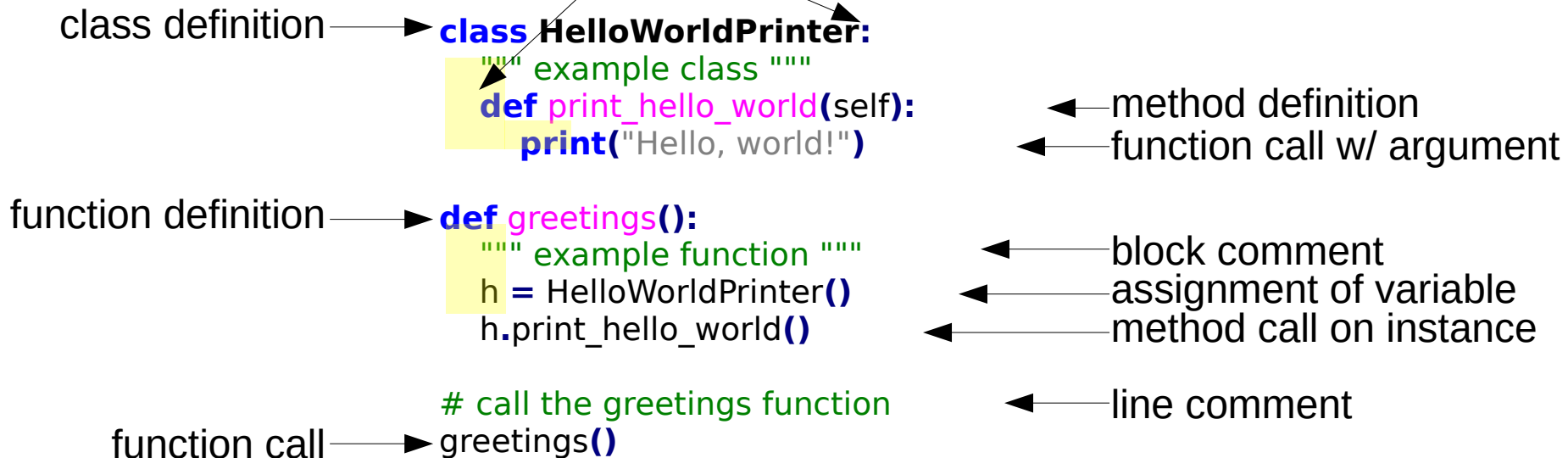
```
class HelloWorldPrinter:
    """ example class """
    def print_hello_world(self):
        print("Hello, world!")

def greetings():
    """ example function """
    h = HelloWorldPrinter()
    h.print_hello_world()

# call the greetings function
greetings()
```


Anatomy of a piece of Python code:

colons and indentation
separate scopes (blocks of code)



Indentation (spaces and tabs at beginning of lines matters!
A scope block is introduced with a colon :
Use # for comments.

Resources: where to get help

- Official Python documentation (actually good):

- <https://docs.python.org/>

Do not forget to switch to your Python version (3.7 in our case).

- The search engine of your choice:

- <https://google.com>
 - <https://duckduckgo.com>
 - <https://swisscows.ch>
 - <https://startpage.com>

- Stack Overflow:

- <https://stackoverflow.com>

- Get help with ? in jupyter-notebook:

- e.g. `print?`



Table Of Contents

- 4. Built-in Types
 - 4.1. Truth Value Testing
 - 4.2. Boolean Operations — `and`, `or`, `not`
 - 4.3. Comparisons
 - 4.4. Numeric Types — `int`, `float`, `complex`
 - 4.4.1. Bitwise Operations on Integer Types
 - 4.4.2. Additional Methods on Integer Types
 - 4.4.3. Additional Methods on Float
 - 4.4.4. Hashing of numeric types
 - 4.5. Iterator Types
 - 4.5.1. Generator Types
 - 4.6. Sequence Types — `list`, `tuple`, `range`
 - 4.6.1. Common Sequence Operations
 - 4.6.2. Immutable

4. Built-in Types

The following sections describe the standard types that are built into the interpreter.

The principal built-in types are numerics, sequences, mappings, classes, instances and exceptions.

Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but `None`.

Some operations are supported by several object types; in particular, practically all objects can be compared, tested for truth value, and converted to a string (with the `repr()` function or the slightly different `str()` function). The latter function is implicitly used when an object is written by the `print()` function.

4.1. Truth Value Testing

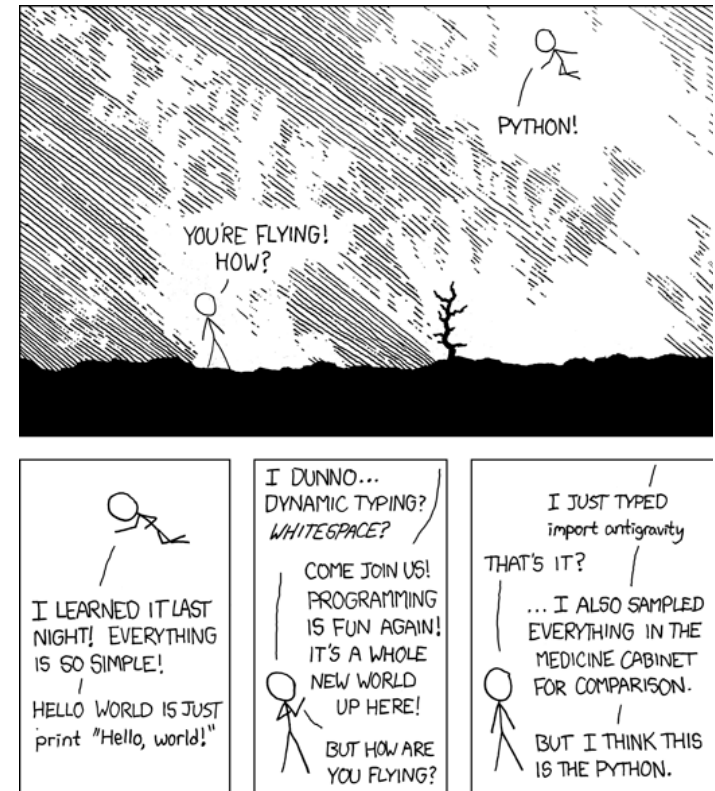
Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations below.

<https://docs.python.org/3.6/library/stdtypes.html>

Python Basics 2 – Notebook

Demo

- Don't forget to activate the Python installation by running anaconda in each new terminal.
- Run `jupyter-notebook` in the Git folder.
- Class material for today is in `Lecture_02_types_conditionals/Lecture_2.ipynb`



<https://www.xkcd.com/353/>

Solve Exercises

Exercise

- Reminder: to get credits for this class you need to show that you have worked seriously on all provided exercises.
- Run jupyter-notebook in your folder.
 - Solve the exercises in `Lecture_02_types_conditionals/Exercise_2.ipynb`
 - Happy solving! The assistants are here to assist you.
- Take a look at the Python and jupyter-notebook cheatsheets in the lecture folder if you want.
- When you are done for today, **commit and push** your changes to your Git repository.