Automating teaching about automation in Python

I heard you like automation, so I put some automation in your automation

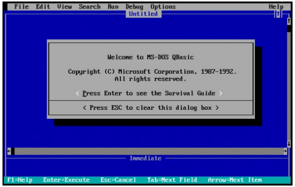
Florian Bruhin

Swiss Python Summit 2022
September 22nd

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST



Welcome to MS-DOS QBasic

Copyright (C) Microsoft Corporation, 1987-1992.
All rights reserved.

< Press Enter to see the Survival Guide >

< Press ESC to clear this dialog box >

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

About me

Florian Bruhin, @The-Compiler

2006	Started programming (QBasic, bash)
2011	Started using Python
2013	Started developing qutebrowser
2015	Switched to pytest, ended up as a maintainer
2016 – 2019	BSc. in Computer Science at ~~HSR~~ OST

40% employed (OST): Teaching Automation with Python to first-semester students
60% open-source and freelancing (Bruhin Software): Python, pytest, Qt

The problem

- Pre-2021: Students learn Java as their primary programming language at OST
- Java can be a pain to deal with[citation needed]
  ...especially if you want a tool to make your life easier rather than learn the fundamentals of programming

The problem

- Pre-2021: Students learn Java as their primary programming language at OST
- Java can be a pain to deal with[citation needed]
  …especially if you want a tool to make your life easier rather than learn the fundamentals of programming

- More and more places where Python is used as a tool
  (to teach math, physics, AI, but also projects, final thesis, etc.)
- Students *demand* learning Python in their studies
  …and lots of schools/universities have introduced/switched to it

The solution

- Fall semester 2021: New course **Automatisierung mit Python** (*Automation with Python*) for **all** first-semester IT students
- In **addition** to Java, but with a **different goal**: Solving real-life problems!
- *"Students will be able to* **use** *the Python programming language [. . .] for simple and complex automation tasks."*

The solution

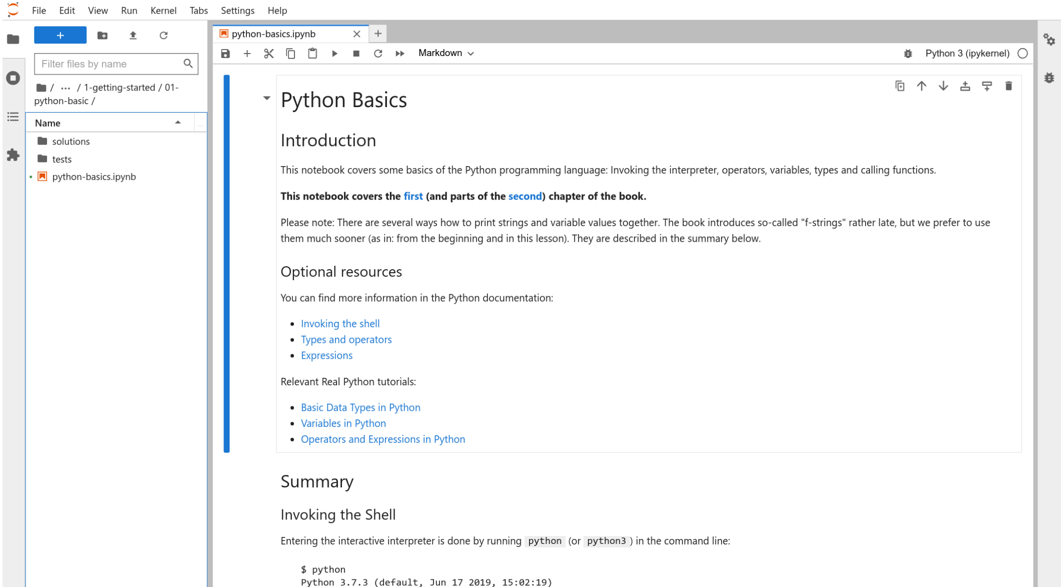
- Fall semester 2021: New course **Automatisierung mit Python** (*Automation with Python*) for **all** first-semester IT students
- In **addition** to Java, but with a **different goal**: Solving real-life problems!
- *"Students will be able to* **use** *the Python programming language [. . .] for simple and complex automation tasks."*

- **Flipped classroom:** No lectures, no paper exam.
  Interactive graded labs and a small graded project!
- We have many newcomers studying IT, or people mostly doing support/network/. . . , without much programming experience.

You need to **get your hands dirty** to learn programming.
It's not just theory, but also a **"craft"**!
We want students to learn **both**: University of **Applied** sciences!

Interactive learning

Labs



Python Basics

Introduction

This notebook covers some basics of the Python programming language: Invoking the interpreter, operators, variables, types and calling functions.

This notebook covers the first (and parts of the second) chapter of the book.

Please note: There are several ways how to print strings and variable values together. The book introduces so-called "f-strings" rather late, but we prefer to use them much sooner (as in: from the beginning and in this lesson). They are described in the summary below.

Optional resources

You can find more information in the Python documentation:

- Invoking the shell
- Types and operators
- Expressions

Relevant Real Python tutorials:

- Basic Data Types in Python
- Variables in Python
- Operators and Expressions in Python

Summary

Invoking the Shell

Entering the interactive interpreter is done by running `python` (or `python3`) in the command line:

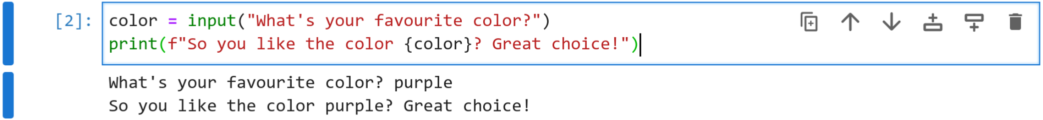
```
$ python
Python 3.7.3 (default, Jun 17 2019, 15:02:19)
```

Interactive learning

Exercises

e) Input and output

- Ask the user for their favourite color and save the result in `color`. Note that it's possible to pass an additional text to be shown to the user (a "prompt") to the function you'll need to use, so you don't need a separate `print` for this.
- Show the text `So you like the color red? Great choice!`, but with `red` replaced by the user's input.

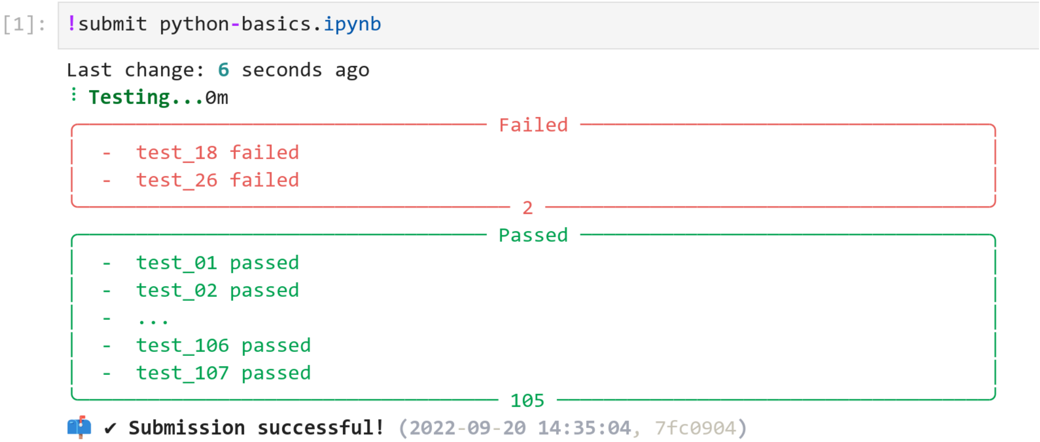


```python
color = input("What's your favourite color?")
print(f"So you like the color {color}? Great choice!")
```

```
What's your favourite color? purple
So you like the color purple? Great choice!
```

Interactive learning

Tests with testbook



```
[1]: !submit python-basics.ipynb
```

```
Last change: 6 seconds ago
⁝ Testing...0m
╭──────────────────────────── Failed ────────────────────────────╮
│                                                                │
│  -  test_18 failed                                             │
│  -  test_26 failed                                             │
╰────────────────────────────── 2 ───────────────────────────────╯
╭──────────────────────────── Passed ────────────────────────────╮
│  -  test_01 passed                                             │
│  -  test_02 passed                                             │
│  -  ...                                                        │
│  -  test_106 passed                                            │
│  -  test_107 passed                                            │
╰───────────────────────────── 105 ──────────────────────────────╯
📫 ✓ Submission successful! (2022-09-20 14:35:04, 7fc0904)
```

Interactive learning

Grading

- 1 ungraded lab (setup and getting started)
- 5 graded labs, 1/3 of final grade, **automated tests**
- Final project, **2/3** of final grade, **graded manually**
  - Python basics, flow control, data structures, . . .
  - Writing a CLI
  - Using web APIs

The… problem?

Over **120 students**, a total of 9 slots (4 hours each) every 2 weeks.

Slightly less this year: 110 or so, and “only” 7 slots.

That’s **a lot** given that I’m doing this the first time!

Thanks, Stefan Richter, for trusting that we could pull it off.

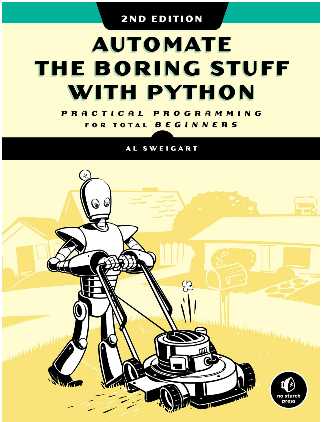
In addition:

- I **love** writing opensource (qutebrowser/pytest), and giving company trainings
- Thus, this needs to stay a **40%** occupation (averaged over a year, I don’t teach in spring)
- Other people are busy too! But I got some help.

  Thanks, Marco, Méline and Urs!

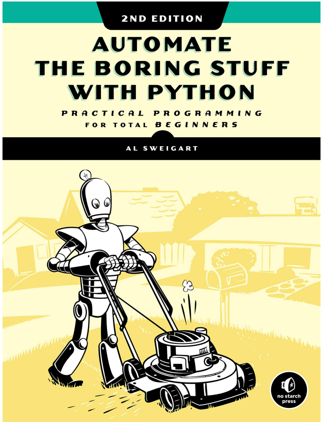
Focusing my attention

- With > 100 students, any kind of manual work with $\mathcal{O}(n)$ is almost certainly **worth automating!**
- I teach students how to make their studies easier. Might as well make **my** job easier!
- I want to focus on the **interesting** part: Creating an environment to help people learn, helping people who are stuck, the beauty of **teaching**.

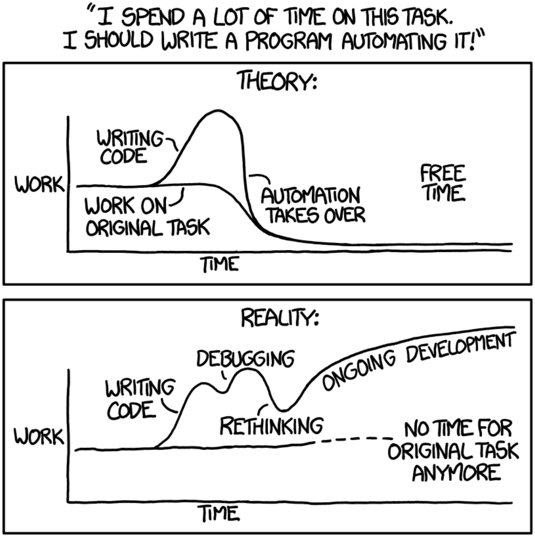

2ND EDITION
AUTOMATE THE BORING STUFF WITH PYTHON
PRACTICAL PROGRAMMING FOR TOTAL BEGINNERS
AL SWEIGART
no starch press

Focusing my attention

- With $>$ 100 students, any kind of manual work with $\mathcal{O}(n)$ is almost certainly **worth automating!**
- I teach students how to make their studies easier. Might as well make **my** job easier!
- I want to focus on the **interesting** part: Creating an environment to help people learn, helping people who are stuck, the beauty of **teaching**.
- Let Python do the **boring** part. Bonus points: It gets easier every year, because more is automated!
- A word of caution: Automation is **not** a substitute for teaching. **Know where to stop!**
- I'm not lazy (...sometimes) – but I want to **focus my attention** on things which benefit students most.


2ND EDITION
AUTOMATE THE BORING STUFF WITH PYTHON
PRACTICAL PROGRAMMING FOR TOTAL BEGINNERS
AL SWEIGART
no starch press

The danger of automation



XKCD 1319, Randall Munroe / xkcd.com

The *real* danger

Eh, I just need a handful of very simple scripts.

I won't bother setting up...

- ...a proper Python package
- ...type annotations
- ...linters / formatters
- ...tests (Yes, I'm a pytest maintainer. Yes, I'm ashamed.)

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines
“taas”: Testing As A Service	430 lines
“AutPy Commander” GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines
“taas”: Testing As A Service	430 lines
“AutPy Commander” GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines
Getting progress about final project hand-ins	350 lines
Grading final projects, sending feedback	680 lines

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines
“taas”: Testing As A Service	430 lines
“AutPy Commander” GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines
Getting progress about final project hand-ins	350 lines
Grading final projects, sending feedback	680 lines
Checking / resetting to deadlines	180 lines
“pseudotaas”: Rerunning tests locally	300 lines
Calculating final grades	550 lines

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines

“taas”: Testing As A Service	430 lines
“AutPy Commander” GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines

Getting progress about final project hand-ins	350 lines
Grading final projects, sending feedback	680 lines

Checking / resetting to deadlines	180 lines
“pseudotaas”: Rerunning tests locally	300 lines
Calculating final grades	550 lines

45 Python files

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines

“taas”: Testing As A Service	430 lines
“AutPy Commander” GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines

Getting progress about final project hand-ins	350 lines
Grading final projects, sending feedback	680 lines

Checking / resetting to deadlines	180 lines
“pseudotaas”: Rerunning tests locally	300 lines
Calculating final grades	550 lines

45 Python files

6500 lines

5000 LOC (tokei)

not including:
950 lines of solutions
3000 lines tests for labs

The *real* danger: Whooops

Development / Deployment scripts	730 lines
Scraping participants	600 lines
Sending welcome mails / other mail code	370 lines
"taas": Testing As A Service	430 lines
"AutPy Commander" GUI	1300 lines
Checking lab submission progress	480 lines
Misc. utilities for handling data	390 lines
Getting progress about final project hand-ins	350 lines
Grading final projects, sending feedback	680 lines
Checking / resetting to deadlines	180 lines
"pseudotaas": Rerunning tests locally	300 lines
Calculating final grades	550 lines

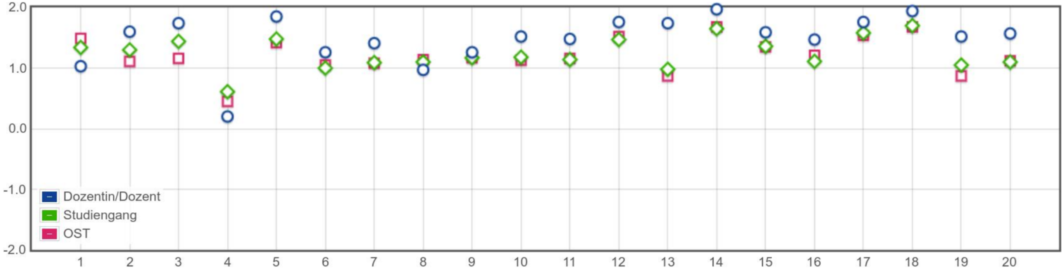
45 Python files

6500 lines

5000 LOC (tokei)

not including:
950 lines of solutions
3000 lines tests for labs

Success!



1. **I was there**
2. **Contents important**
3. **Contents interesting**
4. Needed time is high
5. Content matches desc.
6. **Useful material**
7. Well structured
8. **Understandable**
9. Speed good
10. Extra material / media
11. Link between lect./ex.
12. **Room for questions**
13. **Checking progress**
14. Lecturer competent
15. Link between theory/practice
16. Didactics
17. Lecturer engaged
18. Mutual respect
19. Room / environment
20. **Overall**

Success!

- Concept of “Interactive learing” / “flipped classroom” as a whole.
  Daniele Procida / EvilDMP of Diátaxis[1]:
  “I hardly believe in teaching anymore. The best thing you can do is creating an environment where people learn.”
- Using git as a “database” for student submissions, with one branch per student
- Including test logs (HTML + JSON) in the commits
- Having a custom GUI tool to view a student’s submission and test report
- Project grading based on parsing Markdown checklists
- ...all the other automation really, not regretting any of the time spent on it!

[1] `diataxis.fr`, structuring docs into tutorial/how-to/explanation/reference

Issues

- Students accidentally deleting Jupyter cell tags
  ⇒ Tooling to notify us, protecting cells
- OST GitLab admins migrating storage without making it read-only
  ⇒ Thankfully I had the lost commits locally... please don't do that again!
- Various smaller issues with Jupyter cluster
  ⇒ Ample deadline extension for affected students, we're humans!

Issues

- Students accidentally deleting Jupyter cell tags
  ⇒ Tooling to notify us, protecting cells
- OST GitLab admins migrating storage without making it read-only
  ⇒ Thankfully I had the lost commits locally… please don't do that again!
- Various smaller issues with Jupyter cluster
  ⇒ Ample deadline extension for affected students, we're humans!
- Me forgetting to check some checkboxes in the project grading checklists
  ⇒ Human mistakes bound to happen with so many students and days of grading.
  Caught thanks to detailed feedback mails being sent, additional sanity checks
- Grading system's API is "Download a template `.xlsx`, add grades, upload"
  …but `openpyxl` somehow corrupts template.
  ⇒ Needs further debugging, until then, copy-paste all grades once in Libreoffice

Issues

Things nobody can prepare you for...

Disclaimer: I don't like calling students out for their mistakes, making mistakes is normal.
But those occurrences are just too strange to not tell you about...

Issues

Things nobody can prepare you for...

Disclaimer: I don't like calling students out for their mistakes, making mistakes is normal.
But those occurrences are just too strange to not tell you about...

- Ctrl + C , Ctrl + V

Issues

Things nobody can prepare you for…

Disclaimer: I don’t like calling students out for their mistakes, making mistakes is normal.

But those occurrences are just too strange to not tell you about…

- Ctrl + C , Ctrl + V
- “Allgemeiner Verpeiltheitsfaktor”
  Student was in military (“WK”) for weeks, “didn’t know” they had to hand in stuff

Issues

Things nobody can prepare you for… 🤯

Disclaimer: I don't like calling students out for their mistakes, making mistakes is normal.
But those occurrences are just too strange to not tell you about…

- Ctrl + C , Ctrl + V
- "Allgemeiner Verpeiltheitsfaktor"
  Student was in military ("WK") for weeks, "didn't know" they had to hand in stuff
-
  ```
  r = requests.get("https://random.dog/woof.json")
  data = eval(r.content)
  ```

Issues

Things nobody can prepare you for...

Disclaimer: I don't like calling students out for their mistakes, making mistakes is normal.
But those occurrences are just too strange to not tell you about...

- Ctrl + C , Ctrl + V
- "Allgemeiner Verpeiltheitsfaktor"
  Student was in military ("WK") for weeks, "didn't know" they had to hand in stuff

-
  ```
  r = requests.get("https://random.dog/woof.json")
  data = eval(r.content)
  ```

-
  ```
  git --config user.name "student.email@example.com"
  git --config user.email "password-for-said-email"
  ```

Issues

Things nobody can prepare you for…

Disclaimer: I don’t like calling students out for their mistakes, making mistakes is normal.
But those occurrences are just too strange to not tell you about…

- Ctrl + C , Ctrl + V
- “Allgemeiner Verpeiltheitsfaktor”
  Student was in military (“WK”) for weeks, “didn’t know” they had to hand in stuff

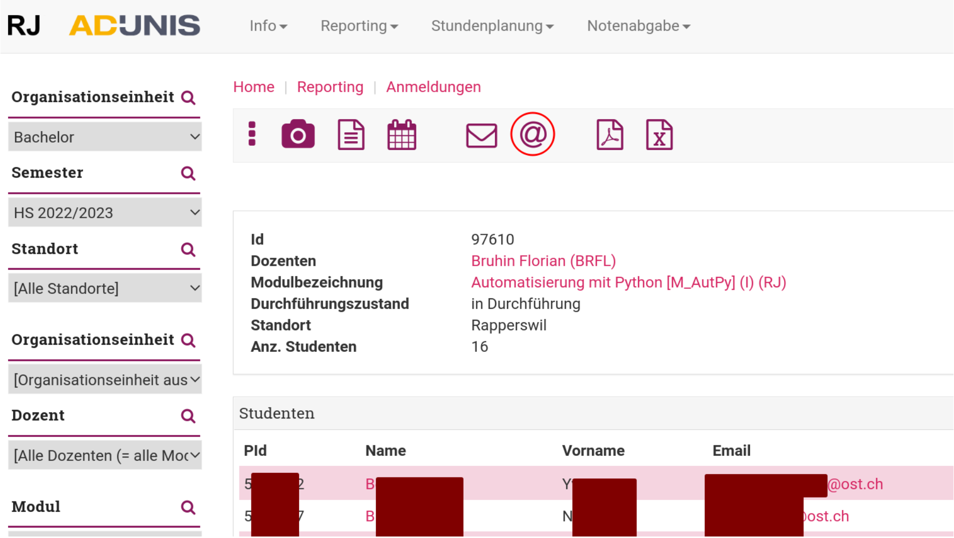
-
  ```
  r = requests.get("https://random.dog/woof.json")
  data = eval(r.content)
  ```

-
  ```
  git --config user.name "student.email@example.com"
  git --config user.email "password-for-said-email"
  ```

⇒ With >100 students, prepare to see **every corner case** you can think of,
and some you’d never think of. Automation **won’t help you** take difficult decisions.

Automation examples

Before the semester



Home | Reporting | Anmeldungen

Organisationseinheit: Bachelor

Semester: HS 2022/2023

Standort: [Alle Standorte]

Organisationseinheit: [Organisationseinheit aus

Dozent: [Alle Dozenten (= alle Moc

Modul

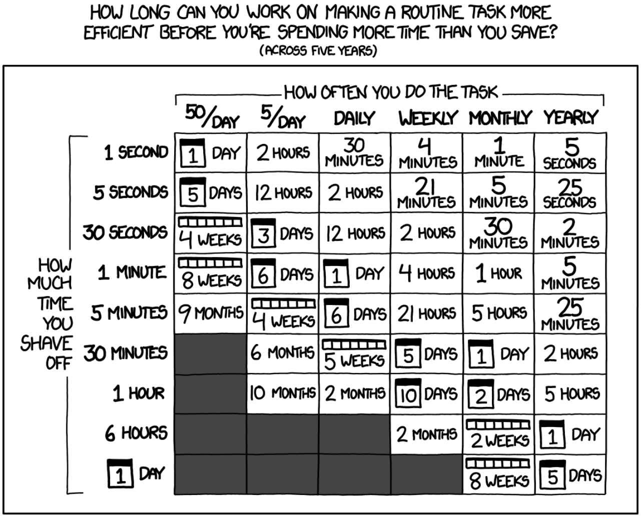
Id	97610
Dozenten	Bruhin Florian (BRFL)
Modulbezeichnung	Automatisierung mit Python [M_AutPy] (I) (RJ)
Durchführungszustand	in Durchführung
Standort	Rapperswil
Anz. Studenten	16

Studenten

PId	Name	Vorname	Email
5[redacted]2	B[redacted]	Y[redacted]	[redacted]@ost.ch
5[redacted]7	B[redacted]	N[redacted]	[redacted]ost.ch

Automation examples

Before the semester



HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?

(ACROSS FIVE YEARS)

HOW OFTEN YOU DO THE TASK

HOW MUCH TIME YOU SHAVE OFF

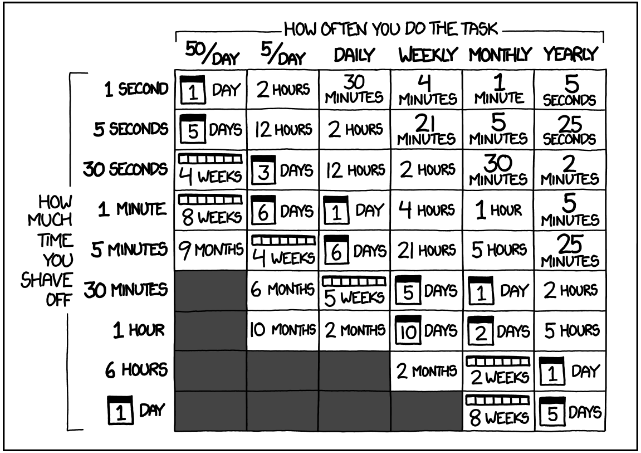
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

XKCD 1205, Randall Munroe / xkcd.com

Automation examples

Before the semester

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)



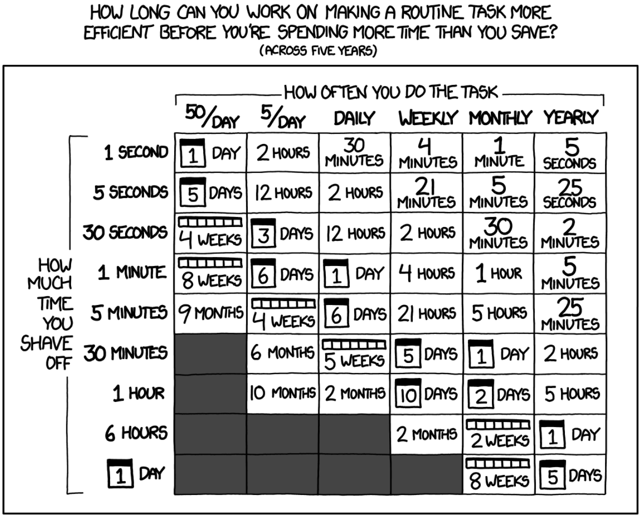
	HOW OFTEN YOU DO THE TASK					
HOW MUCH TIME YOU SHAVE OFF	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	▨	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	▨	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	▨	▨	▨	2 MONTHS	2 WEEKS	1 DAY
1 DAY	▨	▨	▨	▨	8 WEEKS	5 DAYS

XKCD 1205, Randall Munroe / xkcd.com

No automation needed?

Automation examples

Before the semester



HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY					8 WEEKS	5 DAYS

XKCD 1205, Randall Munroe / xkcd.com

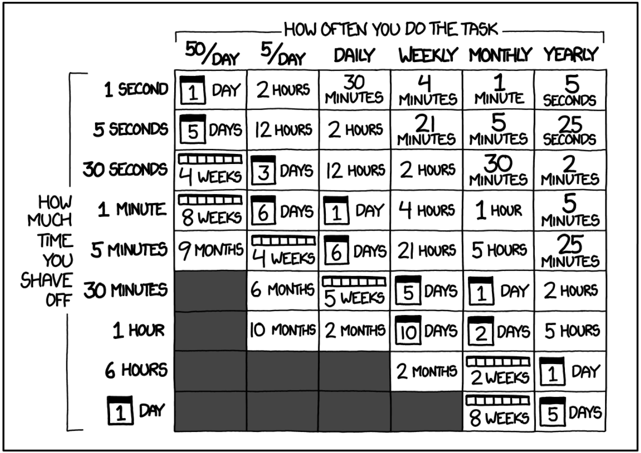
No automation needed?

- People get added last-minute
- ...after preparing
- ...even after semester started

Automation examples

Before the semester

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)



HOW MUCH TIME YOU SHAVE OFF	HOW OFTEN YOU DO THE TASK					
	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

XKCD 1205, Randall Munroe / xkcd.com

No automation needed?

- People get added last-minute
- ...after preparing
- ...even after semester started

- People leave in the middle of the semester
- ...and nobody tells you

Automation examples

Before the semester

- No API (as far as I know)
- Lots of data (200 KB of JSON) in `window.adunisModel = ...`
- Not what we need, however...
- But HTML is structured enough. `requests` and `bs4` to the rescue!

Automation examples

Before the semester

- No API (as far as I know)
- Lots of data (200 KB of JSON) in `window.adunisModel = ...`
- Not what we need, however...
- But HTML is structured enough. `requests` and `bs4` to the rescue!

- Weird Microsoft-based login flow
- Couldn't figure out how it works, libraries seem to be for APIs only

Automation examples

Before the semester

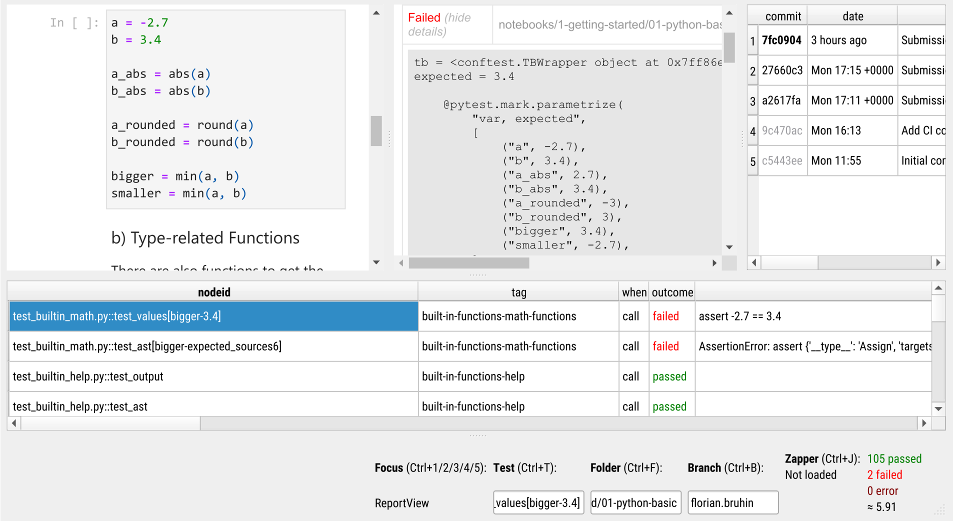
- No API (as far as I know)
- Lots of data (200 KB of JSON) in `window.adunisModel = ...`
- Not what we need, however...
- But HTML is structured enough. `requests` and `bs4` to the rescue!

- Weird Microsoft-based login flow
- Couldn't figure out how it works, libraries seem to be for APIs only

- Whatever, I write a browser since 2013, and I can access cookies
- Login via QtWebEngine browser (injected JS to fill values)
- Grab session cookie, feed it to `requests` 😎

Automation examples

During the semester: Commander



```
In [ ]: a = -2.7
        b = 3.4

        a_abs = abs(a)
        b_abs = abs(b)

        a_rounded = round(a)
        b_rounded = round(b)

        bigger = min(a, b)
        smaller = min(a, b)
```

b) Type-related Functions

There are also functions to get the

Failed *(hide details)* notebooks/1-getting-started/01-python-bas

```
tb = <conftest.TBWrapper object at 0x7ff86e
expected = 3.4

    @pytest.mark.parametrize(
        "var, expected",
        [
            ("a", -2.7),
            ("b", 3.4),
            ("a_abs", 2.7),
            ("b_abs", 3.4),
            ("a_rounded", -3),
            ("b_rounded", 3),
            ("bigger", 3.4),
            ("smaller", -2.7),
```

	commit	date	
1	**7fc0904**	3 hours ago	Submissi
2	27660c3	Mon 17:15 +0000	Submissi
3	a2617fa	Mon 17:11 +0000	Submissi
4	9c470ac	Mon 16:13	Add CI cc
5	c5443ee	Mon 11:55	Initial cor

nodeid	tag	when	outcome	
test_builtin_math.py::test_values[bigger-3.4]	built-in-functions-math-functions	call	failed	assert -2.7 == 3.4
test_builtin_math.py::test_ast[bigger-expected_sources6]	built-in-functions-math-functions	call	failed	AssertionError: assert {'__type__': 'Assign', 'targets
test_builtin_help.py::test_output	built-in-functions-help	call	passed	
test_builtin_help.py::test_ast	built-in-functions-help	call	passed	

Focus (Ctrl+1/2/3/4/5): ReportView

Test (Ctrl+T): values[bigger-3.4]

Folder (Ctrl+F): d/01-python-basic

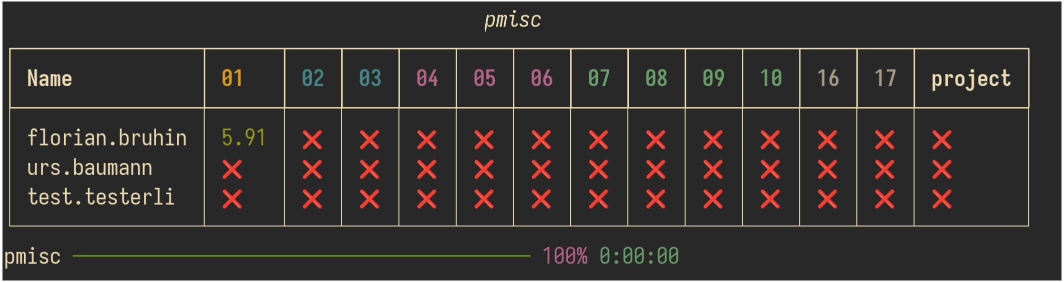
Branch (Ctrl+B): florian.bruhin

Zapper (Ctrl+J): Not loaded

105 passed
2 failed
0 error
≈ 5.91

Automation examples

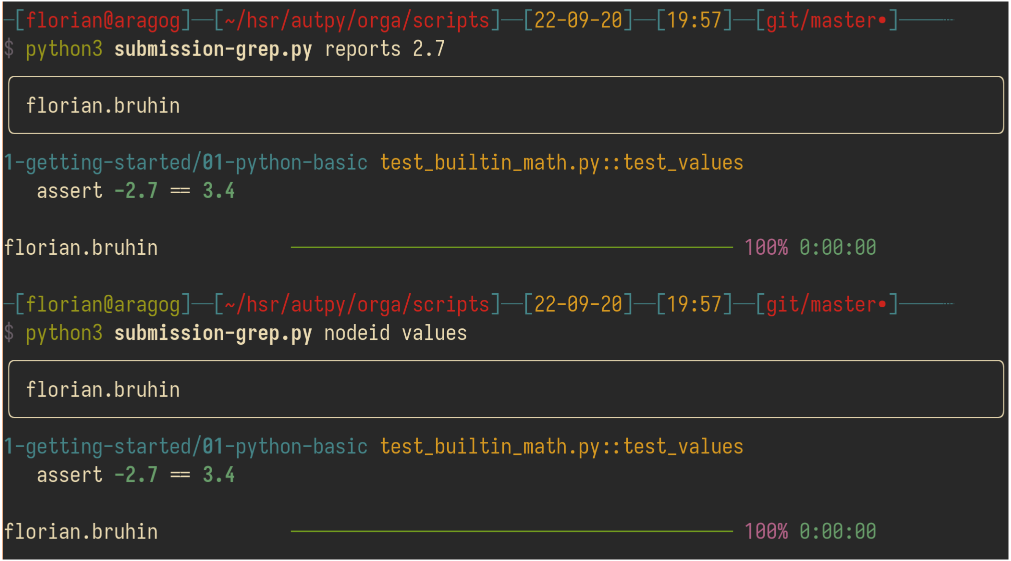
During the semester: Overview



```
                                  pmisc
┌────────────────┬──────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬────┬─────────┐
│ Name           │ 01   │ 02 │ 03 │ 04 │ 05 │ 06 │ 07 │ 08 │ 09 │ 10 │ 16 │ 17 │ project │
├────────────────┼──────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼────┼─────────┤
│ florian.bruhin │ 5.91 │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌      │
│ urs.baumann    │ ❌   │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌      │
│ test.testerli  │ ❌   │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌ │ ❌      │
└────────────────┴──────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴────┴─────────┘
pmisc ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Automation examples

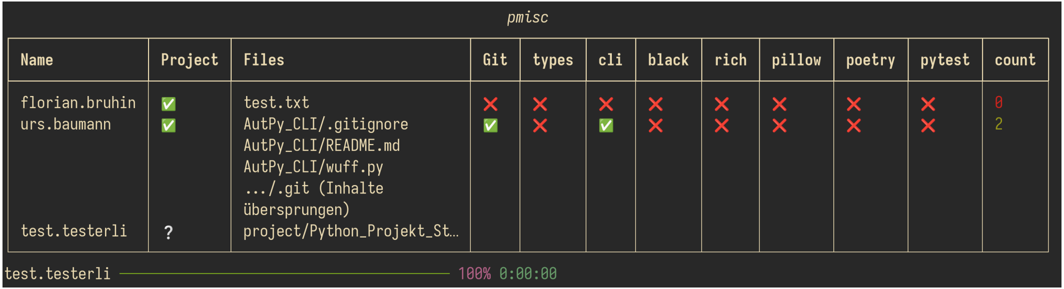
During the semester: Grep



```
─[florian@aragog]─[~/hsr/autpy/orga/scripts]─[22-09-20]─[19:57]─[git/master•]───
$ python3 submission-grep.py reports 2.7

╭──────────────────────────────────────────────────────────────────────────────╮
│ florian.bruhin                                                               │
╰──────────────────────────────────────────────────────────────────────────────╯

1-getting-started/01-python-basic test_builtin_math.py::test_values
   assert -2.7 == 3.4

florian.bruhin            ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00

─[florian@aragog]─[~/hsr/autpy/orga/scripts]─[22-09-20]─[19:57]─[git/master•]───
$ python3 submission-grep.py nodeid values

╭──────────────────────────────────────────────────────────────────────────────╮
│ florian.bruhin                                                               │
╰──────────────────────────────────────────────────────────────────────────────╯

1-getting-started/01-python-basic test_builtin_math.py::test_values
   assert -2.7 == 3.4

florian.bruhin            ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Automation examples

Towards end of semester: Project overview



```
                                              pmisc
┌────────────────┬─────────┬───────────────────────────┬─────┬───────┬─────┬───────┬──────┬────────┬────────┬────────┬───────┐
│ Name           │ Project │ Files                     │ Git │ types │ cli │ black │ rich │ pillow │ poetry │ pytest │ count │
├────────────────┼─────────┼───────────────────────────┼─────┼───────┼─────┼───────┼──────┼────────┼────────┼────────┼───────┤
│ florian.bruhin │ ✅      │ test.txt                  │ ❌  │ ❌    │ ❌  │ ❌    │ ❌   │ ❌     │ ❌     │ ❌     │ 0     │
│ urs.baumann    │ ✅      │ AutPy_CLI/.gitignore      │ ✅  │ ❌    │ ✅  │ ❌    │ ❌   │ ❌     │ ❌     │ ❌     │ 2     │
│                │         │ AutPy_CLI/README.md       │     │       │     │       │      │        │        │        │       │
│                │         │ AutPy_CLI/wuff.py         │     │       │     │       │      │        │        │        │       │
│                │         │ .../.git (Inhalte         │     │       │     │       │      │        │        │        │       │
│                │         │ übersprungen)             │     │       │     │       │      │        │        │        │       │
│ test.testerli  │ ?       │ project/Python_Projekt_St…│     │       │     │       │      │        │        │        │       │
└────────────────┴─────────┴───────────────────────────┴─────┴───────┴─────┴───────┴──────┴────────┴────────┴────────┴───────┘
test.testerli ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
```

Automation examples

After end of semester: Project grading

Project grading workflow:

- Pick random student
- Get zip from submissions repository
- Unpack zip in “grading-area” folder

Automation examples

After end of semester: Project grading

Project grading workflow:

- Pick random student
- Get zip from submissions repository
- Unpack zip in “grading-area” folder

- Show overview (file list, detected features)
- Prepare checklist and open in editor
- Wait until editor closed

Automation examples

After end of semester: Project grading

Project grading workflow:

- Pick random student
- Get zip from submissions repository
- Unpack zip in “grading-area” folder

- Show overview (file list, detected features)
- Prepare checklist and open in editor
- Wait until editor closed

- Parse checklist
- Show parsed points and grade, wait for confirmation
- Commit grading file to submissions repository
- ...and during the whole process, only show names as rot13

Automation examples

After end of semester: Project grading

Project grading workflow:

- Pick random student
- Get zip from submissions repository
- Unpack zip in “grading-area” folder

- Show overview (file list, detected features)
- Prepare checklist and open in editor
- Wait until editor closed

- Parse checklist
- Show parsed points and grade, wait for confirmation
- Commit grading file to submissions repository
- ...and during the whole process, only show names as rot13

```
ABCDEFGHIJKLMNOP...
↓            ↑
NOPQRSTUVWXYZABC...
```

Automation examples

After end of semester: Project grading

Project grading workflow:

- Pick random student
- Get zip from submissions repository
- Unpack zip in “grading-area” folder

- Show overview (file list, detected features)
- Prepare checklist and open in editor
- Wait until editor closed

- Parse checklist
- Show parsed points and grade, wait for confirmation
- Commit grading file to submissions repository
- ...and during the whole process, only show names as rot13

ABCDEFGHIJKLMNOP...

↓ ↑

NOPQRSTUVWXYZABC...

florian.bruhin

↓

sybevna.oehuva

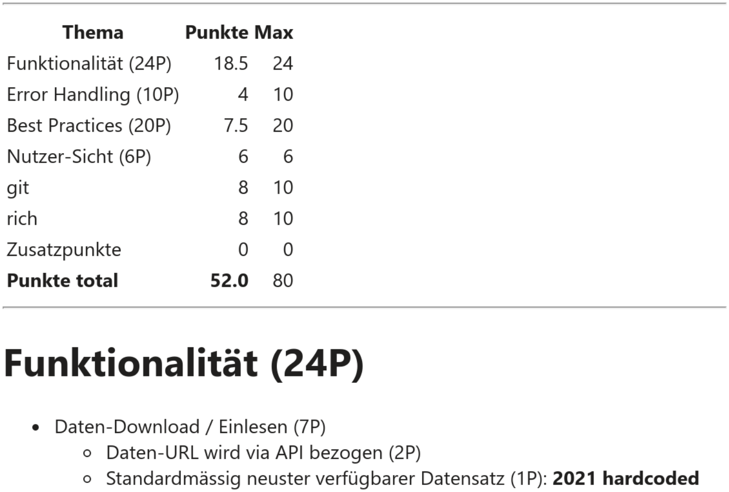
Automation examples

After end of semester: Project grading

```
# Functionality (24P)

- Data download / reading (7P)
  - [ ] Download URL is obtained via API (2P)
  - [ ] Latest available data set used by default (1P)
  - ...

- [ ] Searching for dogs (2P)

- Statistics (9P)
 - [ ] Longest dog name is output correctly (0.5P)
 - [ ] Shortest dog name is output correctly (0.5P)
 - [ ] Top 10 is output correctly (1P)
 - ...
```

Automation examples

After end of semester: Project grading



Thema	Punkte	Max
Funktionalität (24P)	18.5	24
Error Handling (10P)	4	10
Best Practices (20P)	7.5	20
Nutzer-Sicht (6P)	6	6
git	8	10
rich	8	10
Zusatzpunkte	0	0
Punkte total	**52.0**	80

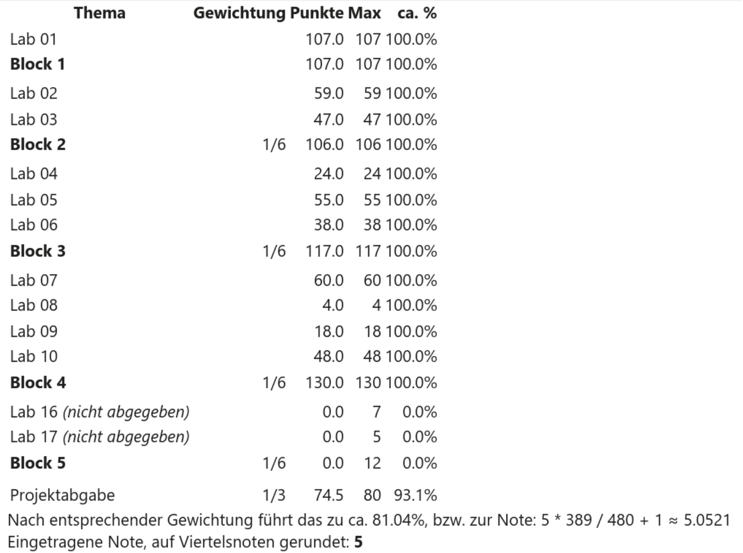
Funktionalität (24P)

- Daten-Download / Einlesen (7P)
  - Daten-URL wird via API bezogen (2P)
  - Standardmässig neuster verfügbarer Datensatz (1P): **2021 hardcoded**

- Parse Markdown checklist
- Calculate points
- Send HTML + plaintext mails

Automation examples

After end of semester: Final grade



Thema	Gewichtung	Punkte	Max	ca. %
Lab 01		107.0	107	100.0%
Block 1		107.0	107	100.0%
Lab 02		59.0	59	100.0%
Lab 03		47.0	47	100.0%
Block 2	1/6	106.0	106	100.0%
Lab 04		24.0	24	100.0%
Lab 05		55.0	55	100.0%
Lab 06		38.0	38	100.0%
Block 3	1/6	117.0	117	100.0%
Lab 07		60.0	60	100.0%
Lab 08		4.0	4	100.0%
Lab 09		18.0	18	100.0%
Lab 10		48.0	48	100.0%
Block 4	1/6	130.0	130	100.0%
Lab 16 *(nicht abgegeben)*		0.0	7	0.0%
Lab 17 *(nicht abgegeben)*		0.0	5	0.0%
Block 5	1/6	0.0	12	0.0%
Projektabgabe	1/3	74.5	80	93.1%

Nach entsprechender Gewichtung führt das zu ca. 81.04%, bzw. zur Note: 5 * 389 / 480 + 1 ≈ 5.0521
Eingetragene Note, auf Viertelsnoten gerundet: **5**

- Rerun all test-cases local, parallelized, 1 Docker container per student
- Calculate final grade Using `fractions`, no rounding!
- Send HTML + plaintext mails

Next steps

- Teaching another ≈ 100 students, with many small improvements!
- Grading another ≈ 100 student projects...

Next steps

- Teaching another ≈ 100 students, with many small improvements!
- Grading another ≈ 100 student projects...

- Turning this into a proper package (`autpy`? `autpypy`? `pyautpy`? `autmetapy`?)
- Type annotations and autoformatters (done!)
- Tests for all the automation logic...
- Using GitPython/pygitops/pygit2/Dulwich/Gittle/... instead of `subprocess` (nicer API and performance)

Next steps

- Teaching another ≈ 100 students, with many small improvements!
- Grading another ≈ 100 student projects...

- Turning this into a proper package (`autpy`? `autpypy`? `pyautpy`? `autmetapy`?)
- Type annotations and autoformatters (done!)
- Tests for all the automation logic...
- Using GitPython/pygitops/pygit2/Dulwich/Gittle/... instead of `subprocess` (nicer API and performance)

- Maybe: Generalizing and publishing?

https://**fstring.help**

https://twitter.com/**the_compiler**
florian@**bruhin.software**