Python in a Week – Conceptual Tests for Learning and Course Development

Christopher Blöcker[1], Thomas Mejtoft[2], **Nina Norgren**[3,4]

[1]Integrated Science Lab, Department of Physics, Umeå University
[2]Department of Applied Physics and Electronics, Umeå University
[3]Department of Molecular Biology, Training Hub, Science for Life Laboratory, Umeå University
[4]Department of Molecular Biology, National Bioinformatics Infrastructure Sweden, Umeå University

TL;DR

Setup

- Short-format (1 week) Python programming course with applications in bioinformatics
- Master and PhD students, Postdocs, early career researchers with little to no programming knowledge
- No summative assessment

Challenge

- Teach useful tools for direct application in (future) job and enable further self-study
- Follow up with students' learning to address misconceptions
- Collect feedback to improve teaching material

Solution

Conceptual tests as short, frequent quizzes for **formative assessment** track learning and **collect feedback** for development.

Programming

- Programming is an essential skill for engineers and scientist, providing foundation to solve complex problems → Ball and Zorn, 2015
- Important part of CDIO Syllabus 3.0, part of fundamental knowledge and reasoning: (2.1) *Analytical reasoning* and (2.2) *Experimentation, investigation, and knowledge discovery*

  → Malmqvist et al., 2022
- Programming language often chosen depending on discipline
- Recently, shift towards Python
  - Simple syntax, easy to learn
  - Industrial relevance
  - Many libraries for statistical analyses, machine learning, visualisations, ...

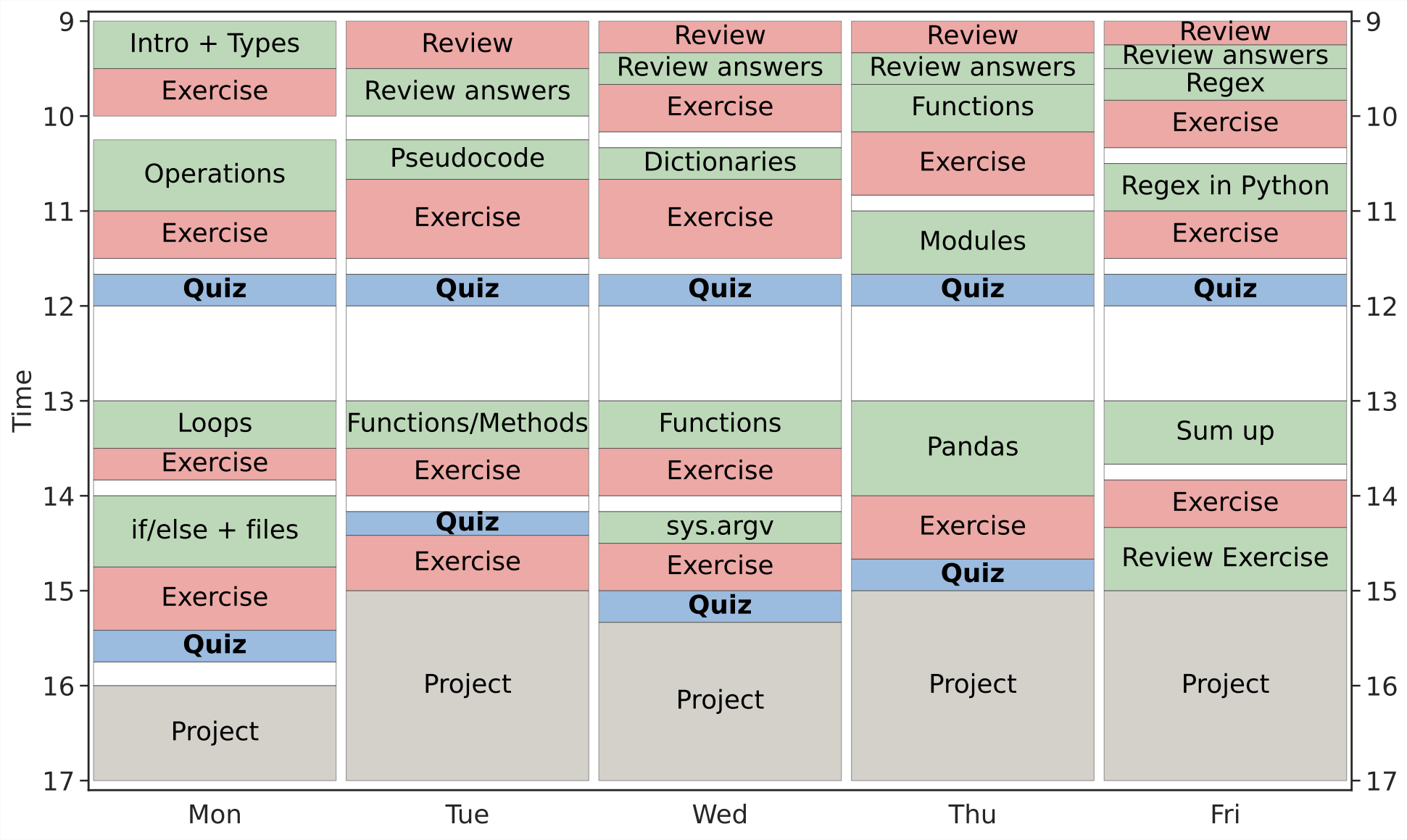
→ Bogdanchikov, Zhaparov, and Suliyev, 2013; Mannila, Peltomäki, and Salakoski, 2006; Jayal et al., 2011; Leping et al., 2009; Cheng, Jayasuriya, and Lim, 2010

Teaching Programming to non-Computer Science Students

- Goal is to teach computational thinking, not educating future programmers: theoretical foundations need not be covered rigorously
  → Vial and Negoita, 2018
- Problems should be selected based on students' discipline
  → Mironova, Vendelin, et al., 2015; Vial and Negoita, 2018
- Typically, teaching non-CS students aims at first-year / early students and courses span a whole semester
  → Cheng, Jayasuriya, and Lim, 2010; Vial and Negoita, 2018; Mironova, Vendelin, et al., 2015; Mironova, Amitan, et al., 2016
- In contrast, our students are already life-science domain experts

Course

Introduction to Python – with Applications in Bioinformatics



Concept Inventories and Misconceptions

- Tools for identifying students' misconceptions through open or closed-ended questions → Taylor et al., 2014
- Well-establised in physics for topics such as force, mechanical waves, electricity and magnetism, ...

  → Hestenes, Wells, and Swackhamer, 1992; Caleon and Subramaniam, 2010; Maloney et al., 2001
- Have also been used in computer science education for teaching Python → Kaczmarczyk et al., 2010; Johnson, McQuistin, and O'Donnell, 2020
- Programming misconceptions can arise due to, for example, differences in the semantics of the same word in programming in natural language, prior math knowledge, flawed mental models regarding how a computer executes code, or inadequate problem-solving strategies

  → Robins, J. Rountree, and N. Rountree, 2003; Qian and Lehman, 2017

Quiz Design

- Designing a concept inventory involves → Goldman et al., 2010
  - setting the scope → learning outcomes
  - identifying misconceptions → own and teaching experience
  - developing questions
  - validation → used quiz in 2022, future work

Design Goals

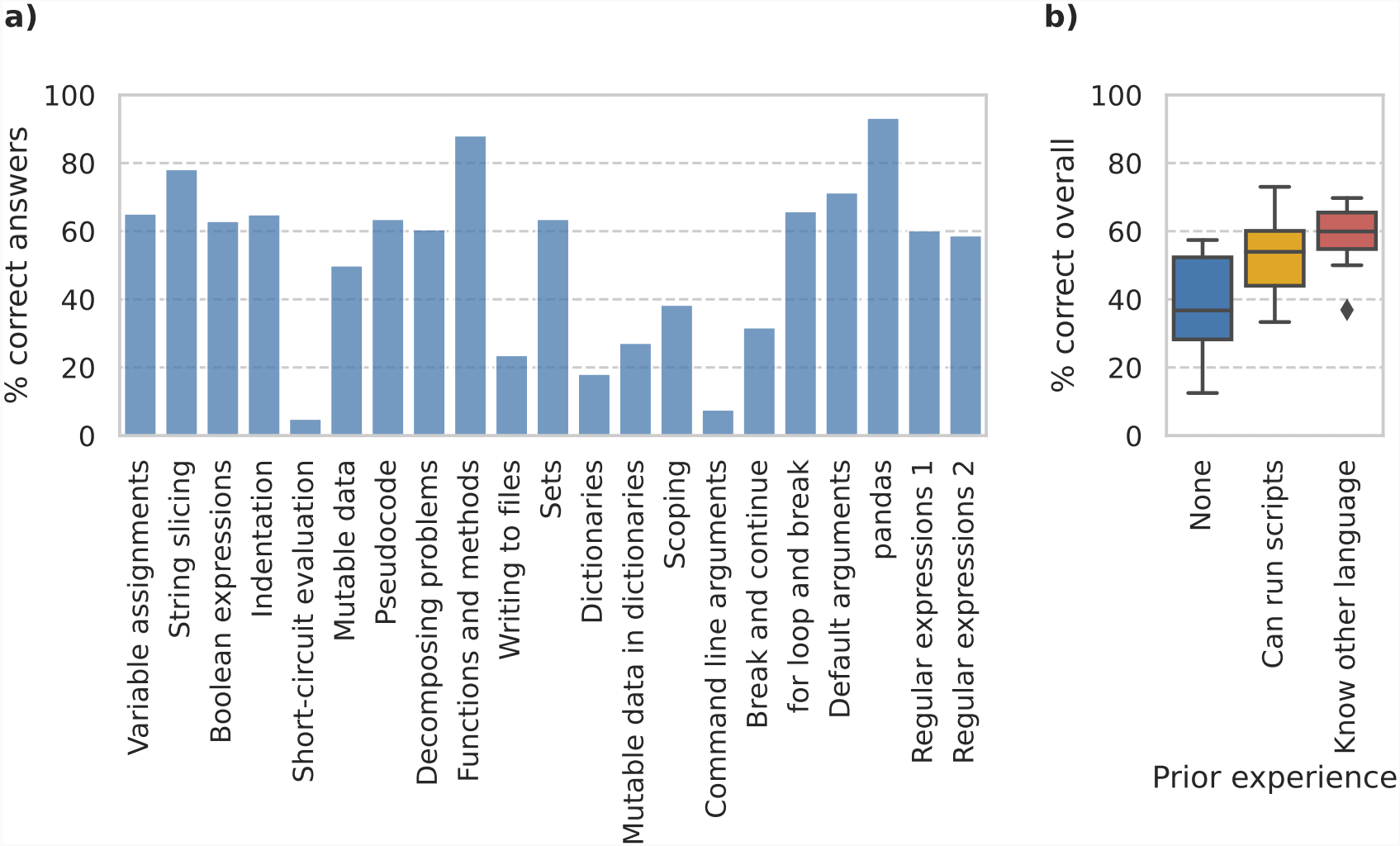
- Test higher-level cognitive abilities: analyse and evaluate

  → Krathwohl, 2002
- Make the quizzes an additional learning activity and confront students with challenges they are likely to face
- Use the quizzes to identify misconceptions

Results

- Conceptual tests containing 21 questions

→ `https://github.com/chrisbloecker/python-in-a-week-quiz`



Results

Variable Scopes

```
x = 1
y = 2

def my_function(x):
    x *= 2
    return x*y

y = 4
z = my_function(2)
print(x,y,z)
```

What does the code print?

(a) 1,2,4
(b) 1,4,8
(c) 1,4,16
(d) 2,4,4
(e) 2,4,8
(f) 2,4,16

Students answered

(a) 0%
(b) 31%
(c) 38%
(d) 0%
(e) 8%
(f) 23%

Short-circuit evaluation

What variable assignment will prevent this code from crashing? Note that not all variables are defined in all cases.

```
if (a and b) or (not a and c) and (d != e):
    print(ok)
```

(a) a=False, b=False, d=False, e=True (10%)
(b) a=True, c=False, d=True, e=False (15%)
(c) a=False, c=False, d=False, e=True (5%)
(d) a=False, c=True, d=False, e=True (70%)

Lessons Learnt

- Take the time to set up the questions, including detailed feedback on both right and wrong answers.
- Emphasise that the quizzes are not a summative assessments, but rather a learning activity. Making mistakes is okay and helps improve conceptual understanding.
- Quizzes were not mandatory, so some students did not answer all questions. Make sure to explain the purpose to the students, and allocate enough time to finish the quizzes.
- Clarify that the quizzes are supposed to be solved by thinking (using pen and paper is ok), but running the code through the Python interpreter to get the right answer defies the purpose.

Conclusion and Future Work

- Conceptual test with 21 questions as a learning activity and several benefits
  - help teachers identify students' misconceptions
  - help students identify their own misconceptions
  - enable timely clarifications
  - provide feedback for course development
- Students report that, despite being challenging, the quizzes were a valuable learning activity that helped them understand programming concepts better
- Future work: long-term follow-up, evaluate and refine our concept inventory

Thank you for your attention!

Questions?