NARA INSTITUTE OF SCIENCE AND TECHNOLOGY
- Outgrow your limits -

大阪大学
OSAKA UNIVERSITY

Universidad
Rey Juan Carlos

# A Method for Inferring Python Proficiency from Textbooks

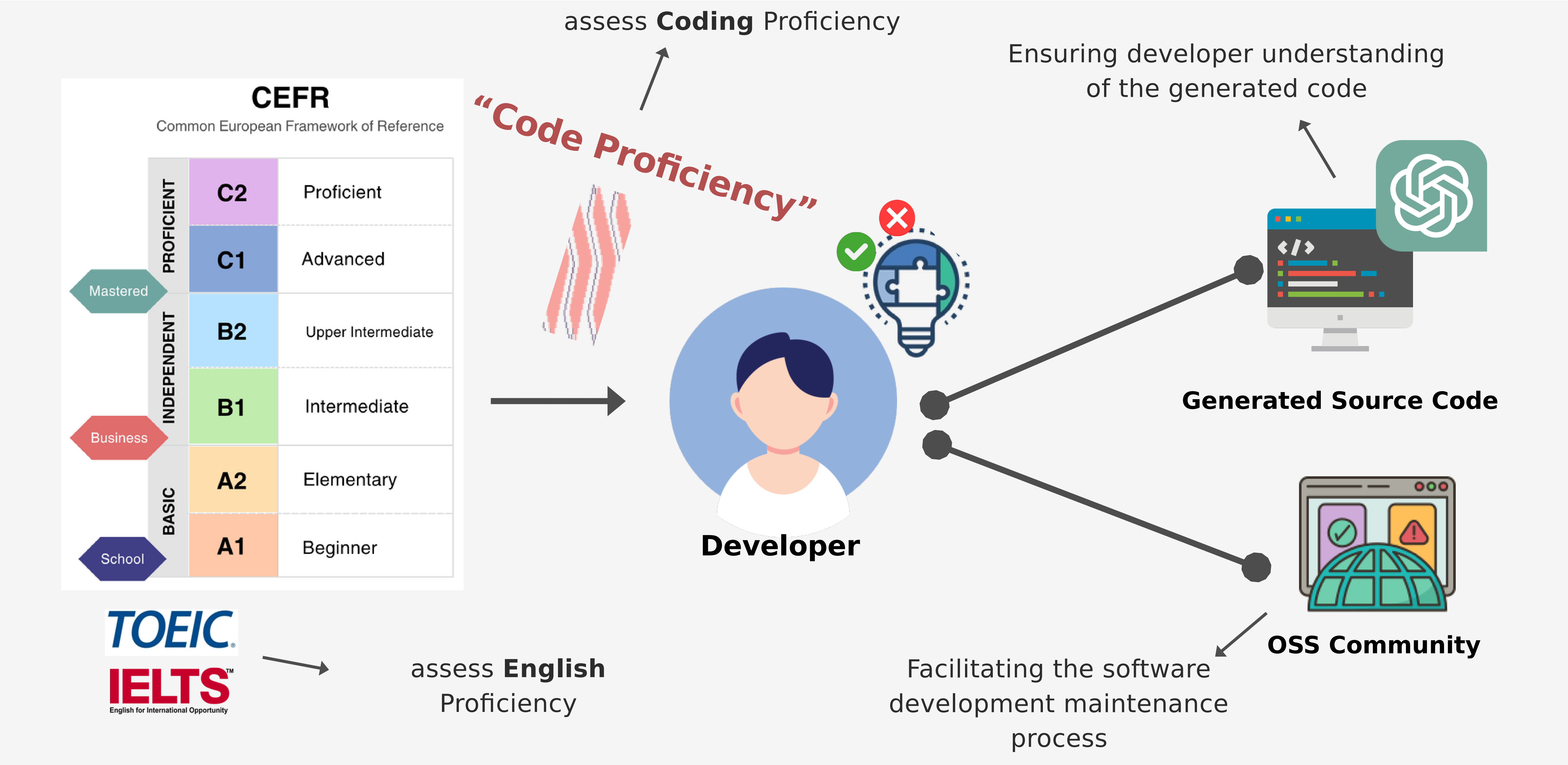
November 17th, BENEVOL 2025, University of Twente

Ruksit Rojpaisarnkit, Gregorio Robles, Jesus M. Gonzalez-Barahona, Kenichi Matsumoto, Raula Gaikovina Kula

[Based, among others, on research published at the ACM TOSEM: https://dl.acm.org/doi/10.1145/3769864]

# Importance of Code Proficiency

**“Code Proficiency reflects an individual’s ability to understand and interpret code constructs.”**



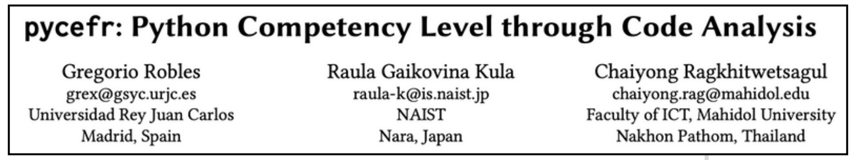
**CEFR**

Common European Framework of Reference

|  |  |  |  |
|---|---|---|---|
| Mastered | PROFICIENT | C2 | Proficient |
|  |  | C1 | Advanced |
| Business | INDEPENDENT | B2 | Upper Intermediate |
|  |  | B1 | Intermediate |
| School | BASIC | A2 | Elementary |
|  |  | A1 | Beginner |

TOEIC
IELTS
English for International Opportunity
assess **English** Proficiency
“Code Proficiency”
assess **Coding** Proficiency
**Developer**
Ensuring developer understanding of the generated code
**Generated Source Code**
**OSS Community**
Facilitating the software development maintenance process

# Determining Code Proficiency is complex



**pycefr: Python Competency Level through Code Analysis**

Gregorio Robles
grex@gsyc.urjc.es
Universidad Rey Juan Carlos
Madrid, Spain

Raula Gaikovina Kula
raula-k@is.naist.jp
NAIST
Nara, Japan

Chaiyong Ragkhitwetsagul
chaiyong.rag@mahidol.edu
Faculty of ICT, Mahidol University
Nakhon Pathom, Thailand

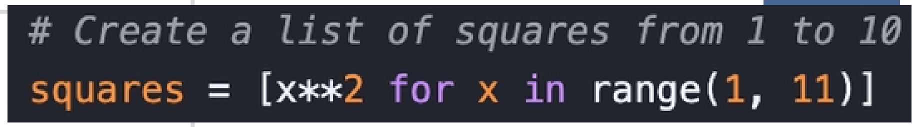
| | | |
|---|---|---|
| PROFICIENT | C2 | zip, map, enumfunc, decaratorfunc |
| | C1 | regex, generatorExpression, dictComp, listComp |
| INDEPENDENT | B2 | __name__, __class__, nestedDictwithList |
| | B1 | while, simpleclass, inheritclass, recursivefunc |
| BASIC | A2 | nestedList, read, write, importfunc |
| | A1 | return, simpleif, forsimple, simpleList, printfunc |

**pycefr Levels [1]**

**we called it**

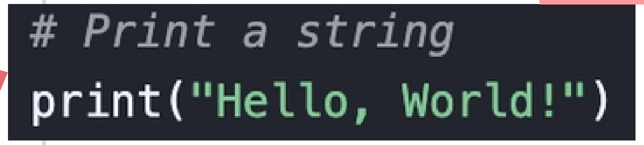
**“code construct”**

C1

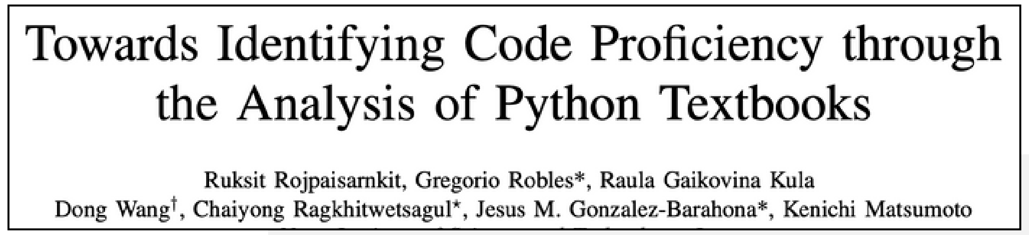


```
# Create a list of squares from 1 to 10
squares = [x**2 for x in range(1, 11)]
```

A1



```
# Print a string
print("Hello, World!")
```



Towards Identifying Code Proficiency through the Analysis of Python Textbooks

Ruksit Rojpaisarnkit, Gregorio Robles*, Raula Gaikovina Kula
Dong Wang†, Chaiyong Ragkhitwetsagul*, Jesus M. Gonzalez-Barahona*, Kenichi Matsumoto

**Previous research**

**Not trivial and highly debatable [2]**

**No standard Proficiency Level**

**This Research**

**Proposing the framework to determine Code Proficiency Levels**

**Textbooks**

**Uber Sequence**

**Clustering**

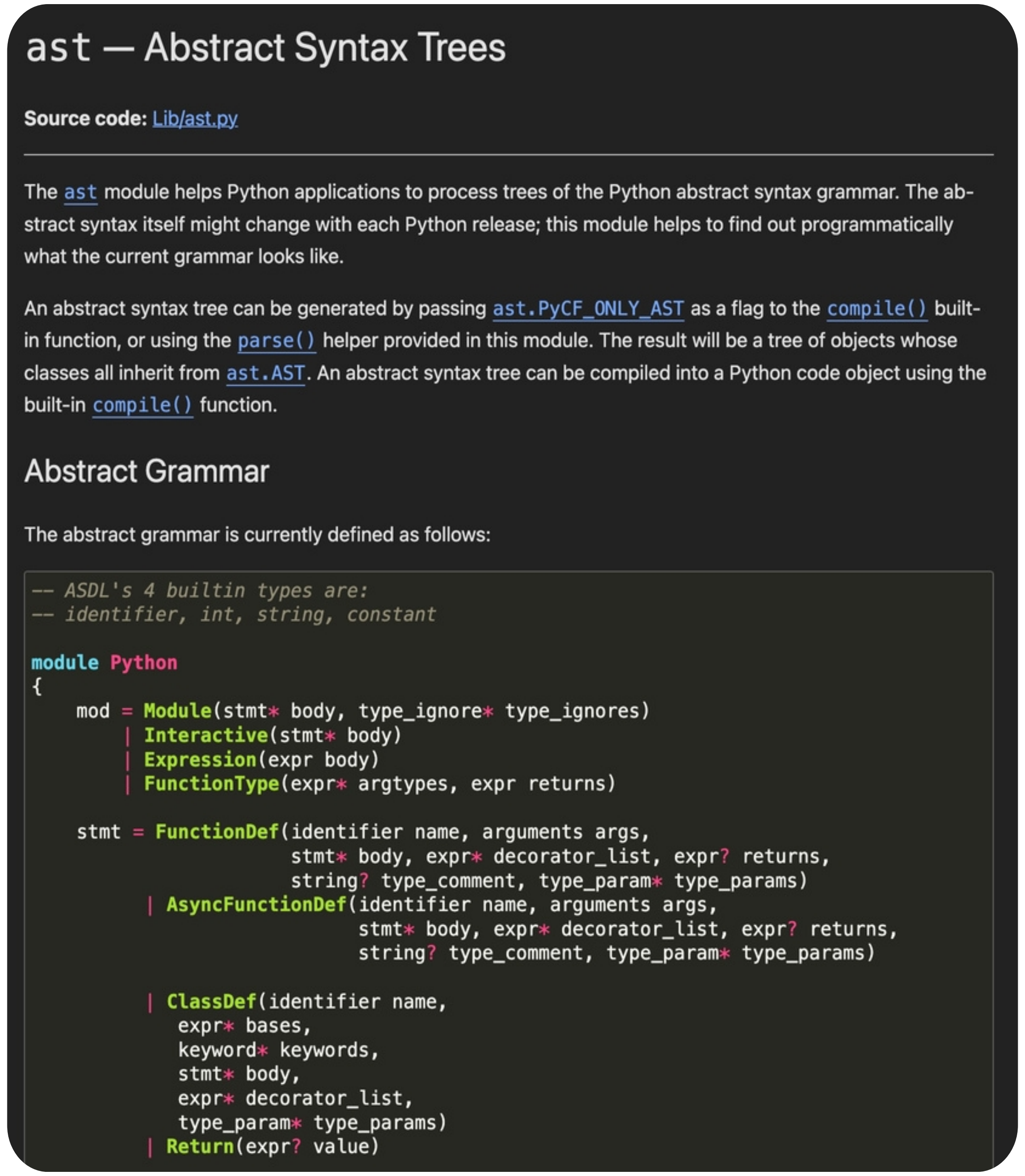
[1] Gregorio Robles, Raula Gaikovina Kula, Chaiyong Ragkhitwetsagul, Tattiya Sakulniwat, Kenichi Matsumoto, and Jesus M. Gonzalez-Barahona. 2022. pycefr: Python Competency Level through Code Analysis. In Proceedings of the 30th IEEE/ACM ICPC

[2] Rojpaisarnkit, R., Robles, G., Kula, R.G., Wang, D., Ragkhitwetsagul, C., Gonzalez-Barahona, J.M. and Matsumoto, K., 2024, October. Towards Identifying Code Proficiency Through the Analysis of Python Textbooks. In 2024 ICSME

# Research Method

Textbooks

## Step 1: Detection Configuration



### ast — Abstract Syntax Trees

**Source code:** Lib/ast.py

The ast module helps Python applications to process trees of the Python abstract syntax grammar. The abstract syntax itself might change with each Python release; this module helps to find out programmatically what the current grammar looks like.

An abstract syntax tree can be generated by passing ast.PyCF_ONLY_AST as a flag to the compile() built-in function, or using the parse() helper provided in this module. The result will be a tree of objects whose classes all inherit from ast.AST. An abstract syntax tree can be compiled into a Python code object using the built-in compile() function.

### Abstract Grammar

The abstract grammar is currently defined as follows:

```
-- ASDL's 4 builtin types are:
-- identifier, int, string, constant

module Python
{
    mod = Module(stmt* body, type_ignore* type_ignores)
        | Interactive(stmt* body)
        | Expression(expr body)
        | FunctionType(expr* argtypes, expr returns)

    stmt = FunctionDef(identifier name, arguments args,
                       stmt* body, expr* decorator_list, expr? returns,
                       string? type_comment, type_param* type_params)
          | AsyncFunctionDef(identifier name, arguments args,
                             stmt* body, expr* decorator_list, expr? returns,
                             string? type_comment, type_param* type_params)

          | ClassDef(identifier name,
             expr* bases,
             keyword* keywords,
             stmt* body,
             expr* decorator_list,
             type_param* type_params)
          | Return(expr? value)
```

Code Construct

read()
readline()
write()
writelines()

print()
return()
simpleassign
() ifelse

/(reg)ex/

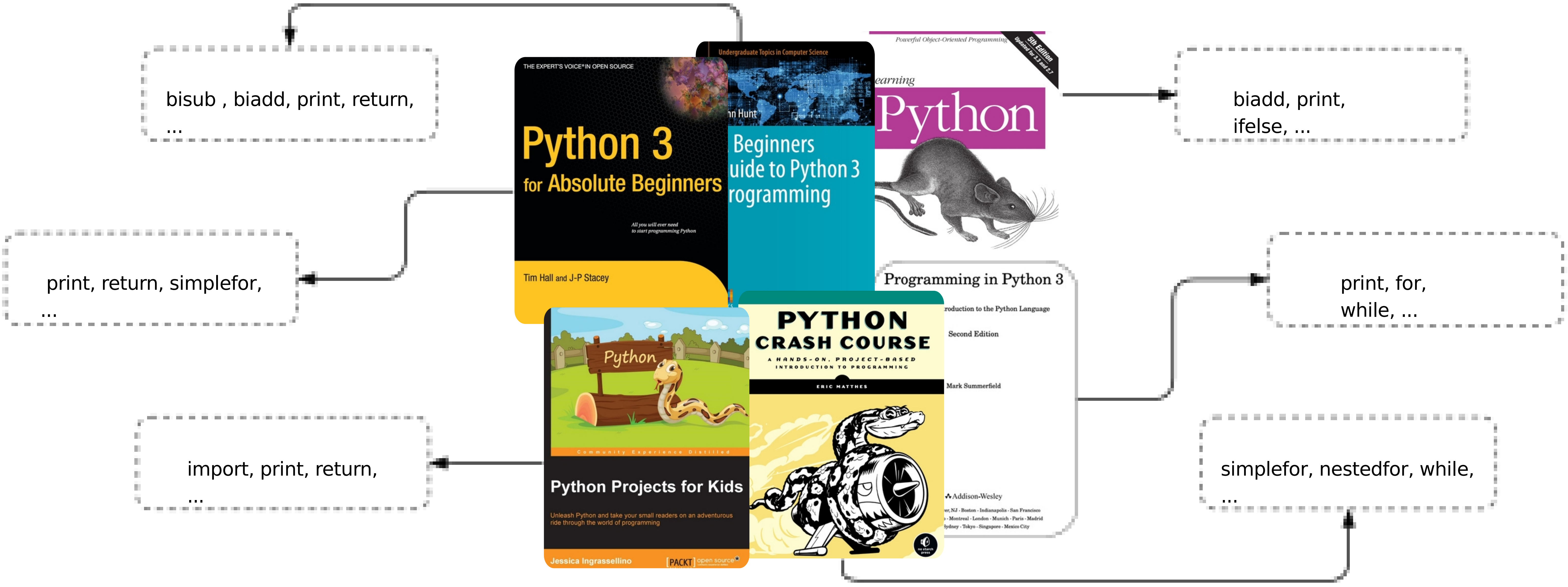
read(): \w+.*.read\(.*\) write(): \
w+.*.write\(.*\) simpleassign: \w+.*\
s*=\s*\w+.* ifelse: if\s+\w+.*:\n.*\
nelse:.*

**Step 1:** Gathering the list of Code Constructs

**Step 2:** Developing the list of Regular Expressions

# Research Method (cont.)

## Step 2: Ordering code constructs



bisub , biadd, print, return, …

print, return, simplefor, …

import, print, return, …

biadd, print, ifelse, …

print, for, while, …

simplefor, nestedfor, while, …

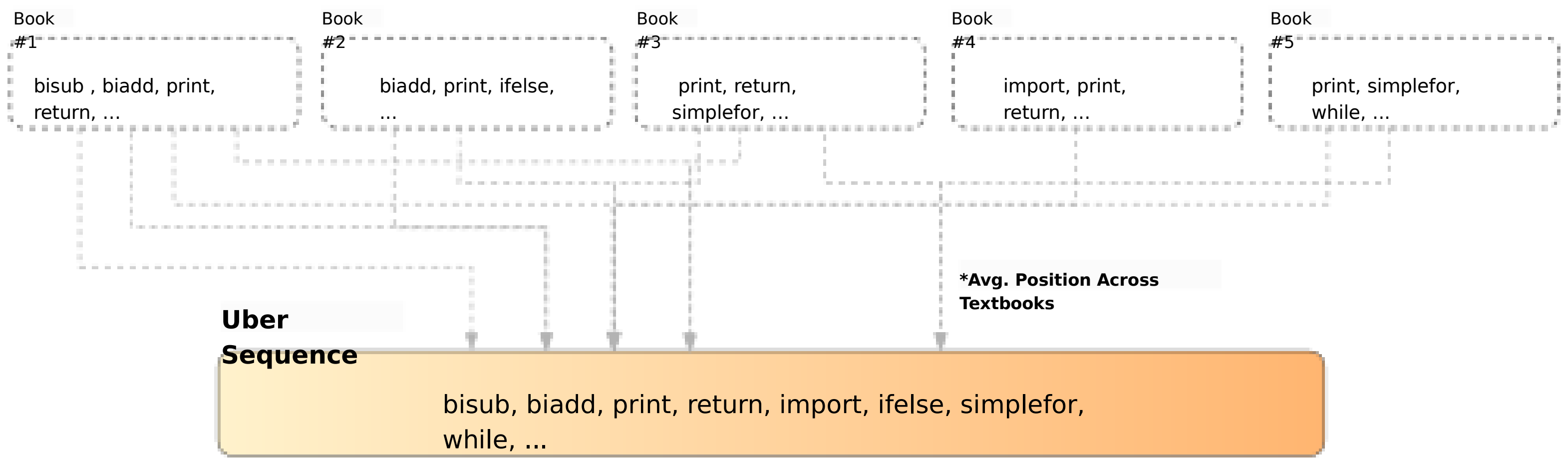
THE EXPERT'S VOICE® IN OPEN SOURCE
Python 3 for Absolute Beginners
Tim Hall and J-P Stacey
Undergraduate Topics in Computer Science
Beginners Guide to Python 3 Programming
Learning Python
5th Edition
Programming in Python 3
Second Edition
Mark Summerfield
Addison-Wesley
Python Projects for Kids
Jessica Ingrassellino
PACKT
PYTHON CRASH COURSE
A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING
ERIC MATTHES

**Extracting sequence of the first appearance of Code Construct in each textbook**

# Research Method (cont.)

## Step 3: Determining Code Proficiency Levels

### Übersequence



Book #1

bisub , biadd, print, return, ...

Book #2

biadd, print, ifelse, ...

Book #3

print, return, simplefor, ...

Book #4

import, print, return, ...

Book #5

print, simplefor, while, ...

*Avg. Position Across Textbooks

Uber Sequence

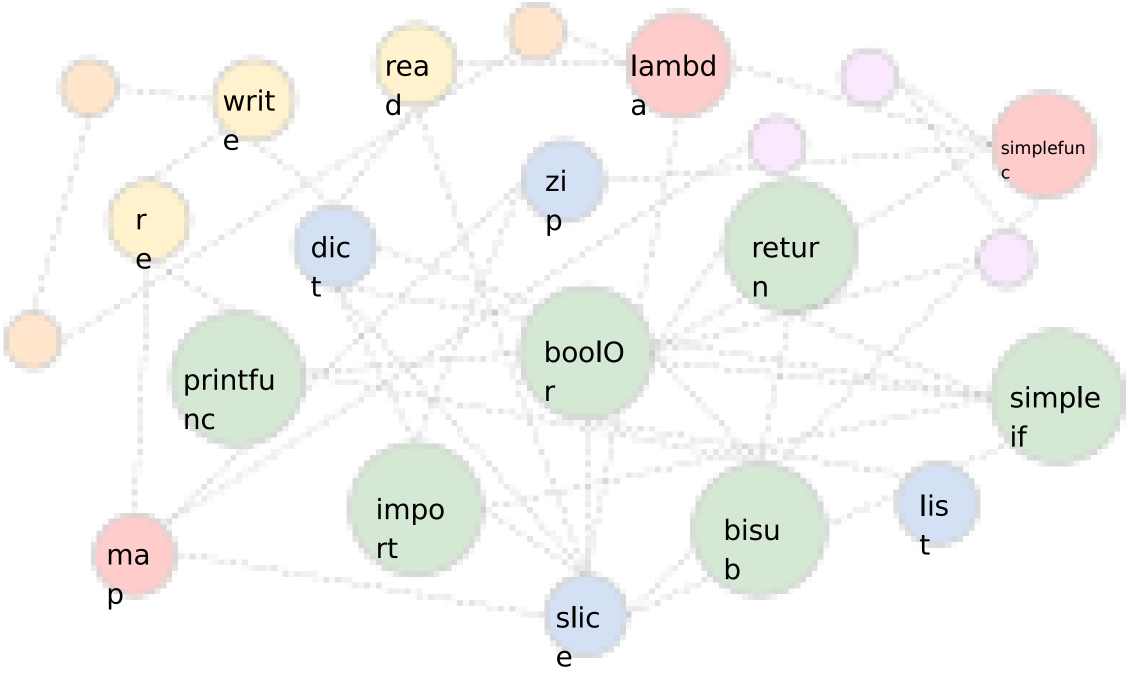
bisub, biadd, print, return, import, ifelse, simplefor, while, ...

**Step 1:** Removing the outlier that exceeds the standard deviation

**Step 2:** Calculating the average position of each Code Construct across textbooks

**Step 3:** Sorting the average position of Code Construct

### Community Clustering



**Node:** Code Construct

**Edge:** Order in the sequence

**Edge Weight:** Position different (page#)

**Algorithm:** Louvain community detection algorithm (100 times)

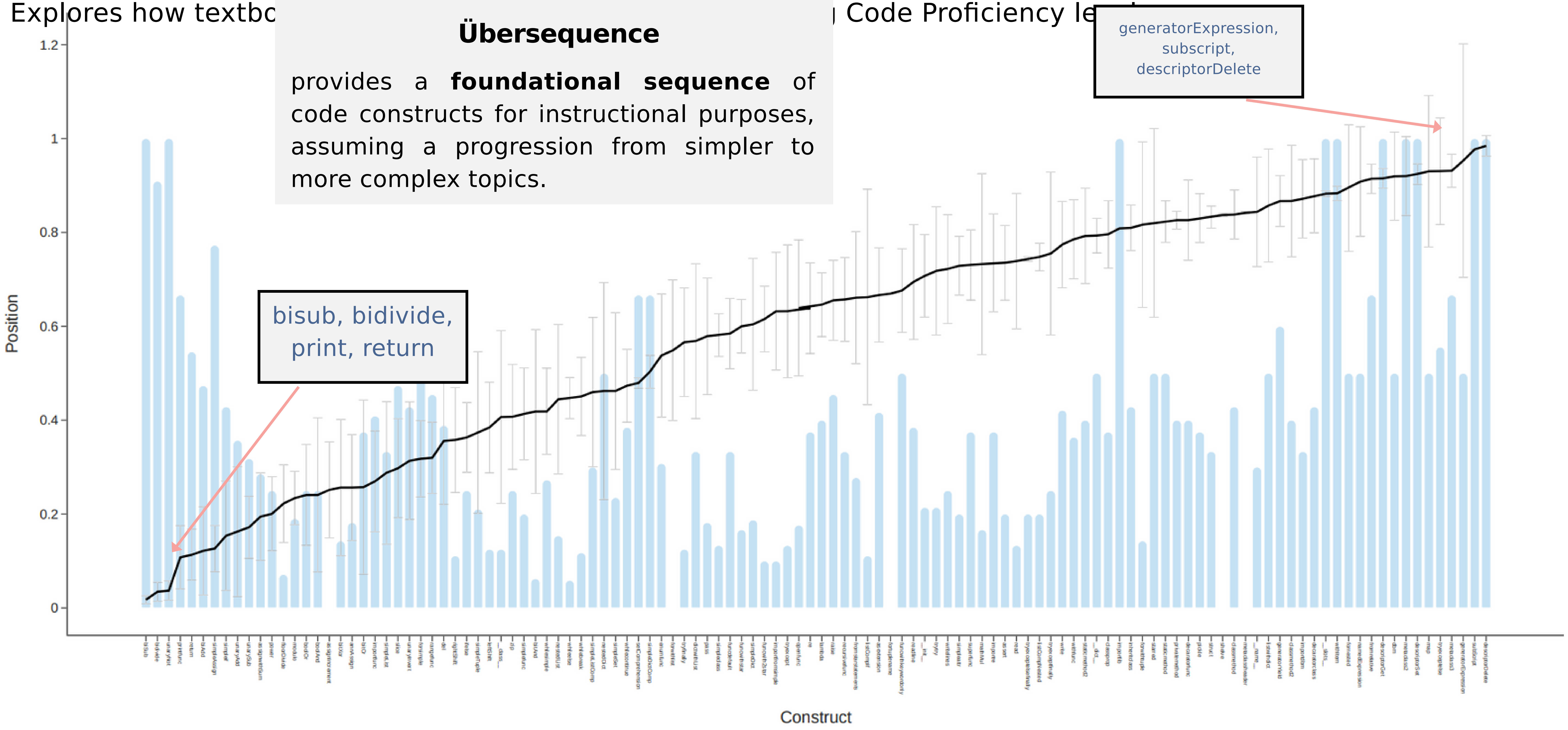
# An Application to Python

## How can textbooks be used to determine Proficiency level?

Explores how textbo … Code Proficiency le

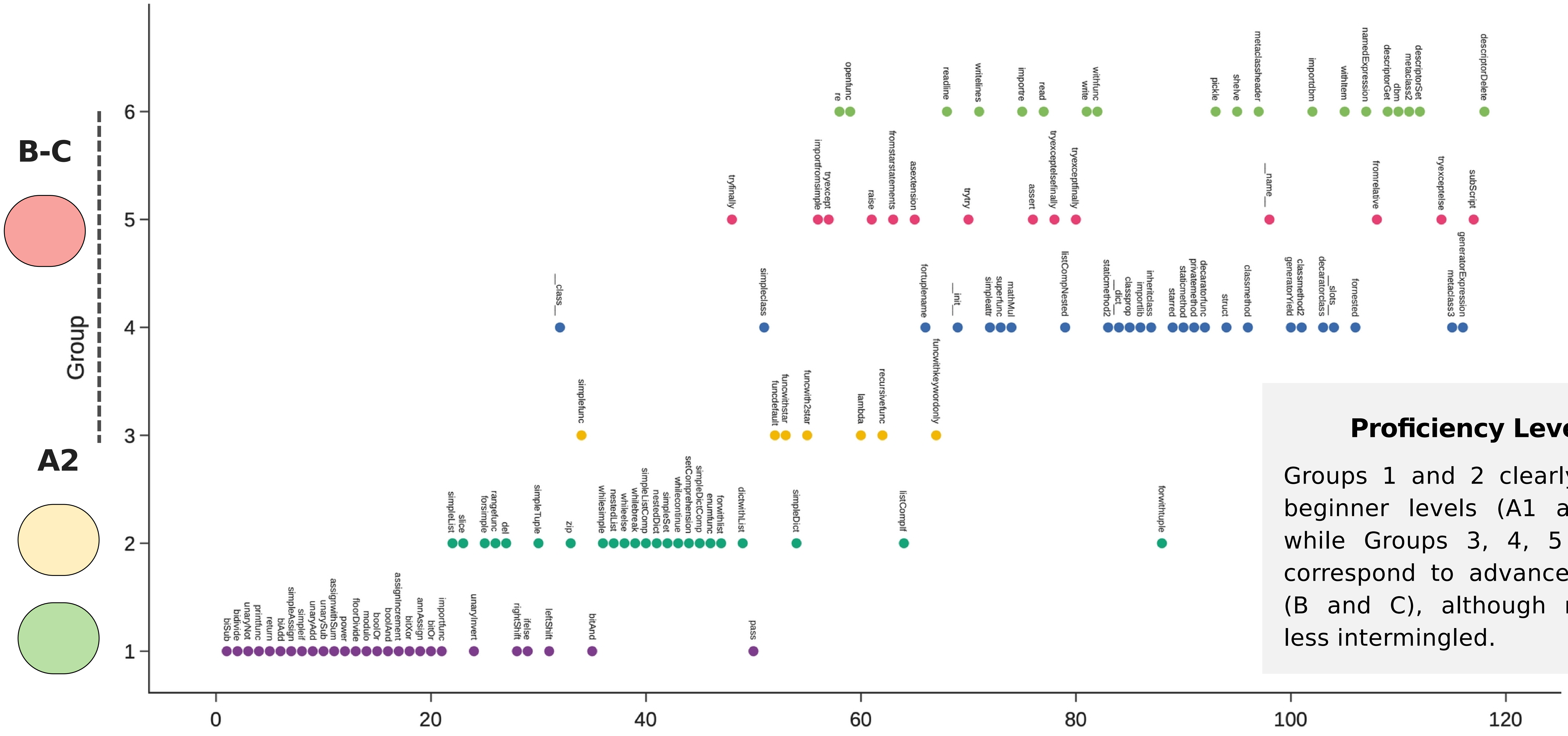
### Übersequence

provides a **foundational sequence** of code constructs for instructional purposes, assuming a progression from simpler to more complex topics.



# An Application to Python (and II)

## Sequence with Community Clustering

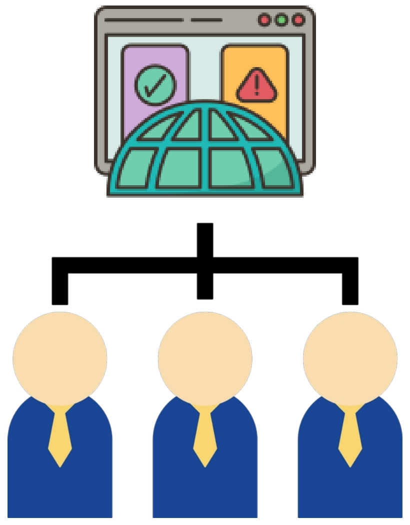


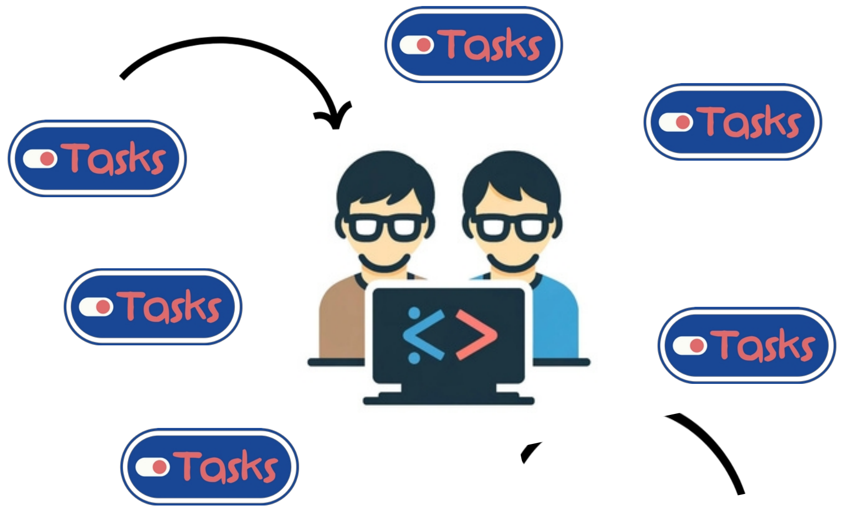
### Proficiency Levels

Groups 1 and 2 clearly match beginner levels (A1 and A2), while Groups 3, 4, 5 and 6) correspond to advanced levels (B and C), although more or less intermingled.

# Limitations and Future Outlook

| | |
|---|---|
| L | There is no clear assignment for higher levels.<br>Are there dialects? |
| F | Another way of characterizing Python code.<br>Idea is transferrable to other programming languages. (To all?) |
| F | Help in code understanding when using AI-assisted code generators?<br>Will we be able to transform code from one level to another one? (e.g., C1 → B1) |





# A Method for Inferring Python Proficiency from Textbooks

November 17th, BENEVOL 2025, University of Twente

Ruksit Rojpaisarnkit, Gregorio Robles, Jesus M. Gonzalez-Barahona, Kenichi Matsumoto, Raula Gaikovina Kula

[Based, among others, on research published at the ACM TOSEM: https://dl.acm.org/doi/10.1145/3769864]