

'Squashing bugs not snakes': Designing a Python toolkit to develop students' programming proficiency

Muhie Al Haimus, Ilanthiraiyan Sivagnanamoorthy,
Yash Rajesh Vaghela

Project Supervisor: Dr. Rehan Shah

Python Toolkit: An Overview

- Why do we need it?
- How was it embedded in a first-year applied mathematics module?
- What did students say and what impact it did it have?
- Concluding remarks and next steps

Attitude towards Programming: I'm doing engineering, why do I need to code?

- Consensus among engineering students is that programming is **particularly difficult**.
- Particularly felt within groups that have **not been accustomed to programming** in earlier stages of their education.
- Programming is traditionally **taught** to engineers **very sparsely**, leading to a lack of conceptual understanding.
- Not explained from the foundations of how a programming language works so it can be **hard to conceptualise**.



Figure 1: Person struggling to debug code [1]

Feedback from past Engineering cohorts

*“In the **Python** section **you are left alone**, and **very little teaching** occurs. Very basic concepts are explained well such as print statements, whilst complex issues like how to use a specific library that are used are not explained as well.”*

*“I think the Python element of the module **needs more contact hours** and needs to be a more guided lesson.”*

*“**More questions** can be given for practice; Python could have a bit **more guidance**, it can be very **hard for those that have not done any coding before**”*

Need for a Python Toolkit in First-Year Engineering

- To bridge the knowledge gap and create a **‘level playing field’** as students enter university with **varying levels of programming experience**
- To **help students** avoid frustration when learning Python by **addressing common errors** that beginners often make
- To **build confidence** in Python by delivering targeted practice questions that **reinforce fundamental concepts**
- To **enhance students’ problem-solving skills** through a **practice-oriented learning approach**, aligning with the **QMUL Graduate Attributes framework**

How was this embedded in the module?

- Co-creation of a Python toolkit by current **undergraduate students** who completed the module within the past two years, prior to the toolkit's implementation
- First pilot run implemented during the academic year **2024/25**, in a first-year undergraduate **applied mathematics module** for engineering students, involving **400 students**
- **Components of toolkit:**
 - Common errors handbook
 - Practice questions booklet
 - Video resources
- Embedded as an **asynchronous, formative resource** alongside course content

 EMS412 Common Errors Toolkit




✓ Done ▾

 EMS412 Practice Questions Toolkit



✓ Done ▾

 Feedback Survey

Opened: Monday, 16 September 2024, 9:00 AM Closes: Friday, 29 November 2024, 5:00 PM



To do ▾

Python Installation Videos

Please also go through the video resources below created by a former student Mr. Muhie Sultan Faisal Al Haimus going over the step-by-step installation of Python for Windows, Mac and Linux operating systems.

 Python Installation Video (Windows)



 Python Installation Video (Mac)



 Python Installation Video (Linux)



EMS412U/EMS506U - Python Common Errors Toolkit:

“Squashing Bugs. Not Snakes.”

Muhie Al Haimus Yash Vaghela Ilanthiraiyan Sivagnanamoorthy

Dr. Rehan Shah

September 2024

Please email m.alhaimus@se23.qmul.ac.uk for any additional comments or corrections for this toolkit.

1 Introduction

Python is an extremely important skill to master and is used extensively throughout your undergraduate course at Queen Mary University of London. Python is an interpreted programming language, which means that each line of code is translated into machine code and executed (run) one at a time. When the interpreter encounters an error, it stops execution and shows an error message along with the specific line where the issue occurred. On the other hand, compiled languages like C/C++ and Java undergo a compilation process where the entire code is converted into machine code all at once, which can present its own set of challenges for beginners. A great explanation of interpreters is available [here](#) (watch up to 1:50) [1].

Understanding how code executes by performing a dry run — where you mentally trace through the code without actually executing it — is a crucial programming skill. This process can be done mentally or by using a provided template (see Section 11). Dry running code helps you examine how the program will behave and identify potential issues before running it.

2 Developing best practices for coding

When learning any new skill it is really important to make sure you don't develop any bad habits. One really important good programming habit is to include comments at the start of your program. These comments **must** include:

- An author
- A date
- A short explanation of the program

Adding comments at the start of your program is helpful for both you and others, as it offers a concise overview of the code's purpose. **In all the subsequent examples you will see this initial commenting style.**

```
NameError                                Traceback (most recent call last)
Cell In[1], line 9
      7 my_cities = ["Jaffna", "London", "Chennai", "Tokyo"]
      8 random.choice(my_cities)
----> 9 print(city)

NameError: name 'city' is not defined
```

Figure 25: Python traceback showing a `NameError` due to the undefined variable reference on line 9

Solution: To fix this, assign the random choice to the `city` variable before the `print` statement. In the following figure, the random selection from `my_cities` is correctly assigned to `city`, allowing the program to run without errors (in this case, producing the output “Jaffna”).

```
# Author: Ilan
# Date: 04/08/2024
'''The program is intended to randomly select a city from
an array of cities, and is to output this random choice.'''

import random
my_cities = ["Jaffna", "London", "Chennai", "Tokyo"]
city = random.choice(my_cities)
print(city)

Jaffna
```

Figure 26: Code snippet of the corrected program, along with the output produced

Note: It's important to follow naming conventions for variables and avoid using Python keywords (e.g. `print`) as identifiers, as this can lead to unexpected behavior and errors in your code.

5.5 Concatenating incompatible data types

Concatenation refers to the process of joining multiple sequence data into a single one, such as joining two strings. Python does not allow implicit conversion between strings and other types for concatenation, so you need to do data conversion before concatenating.

```
# Author: Ilan
# Date: 04/08/2024
'''The program is intended to output a message with a score'''

marks = 70
print("You scored " + marks)
```

Figure 27: Code snippet with a string-integer concatenation error

EMS412U: Python Practice Questions Toolkit

Muhie Al Haimus, Yash Vaghela, Ilanthiraiyan Sivagnanamoorthy, Dr. Rehan Shah

September 2024

Please email m.alhaimus@se23.qmul.ac.uk for any additional comments or corrections for this toolkit.

1 Introduction

This toolkit serves as extra practice for the Python section of EMS412U. This resource should not be solely used for revision and should be used as **extra** practice to get familiar with Python as a programming language for the coursework element of the module. User inputs are *not* covered in the EMS412U module, however, you can make projects using this feature to make them more interesting for you as a programmer and gain a better understanding of what you can do with Python in the future.

2 Practice questions for Weeks 1-6

2.1 Week 1: User input

- 1) Write a program to read the user's name and then display a message to welcome them by name (e.g. “Hello Bob!”).
- 2) Write a program to read in two numbers and print their sum and average. Run your program with the numbers 12 and 15 to show that it works.

2.2 Week 1: Data types

- 1) Write a program to read in the amount of the bill and the amount of money tendered. Calculate and display the amount of change due to the customer. Run your program to show that it works with a bill amount of £8.50 and an amount tendered of £10.
- 2) Write a program to read in the amount of a restaurant bill and then ask the user how much as a percentage of the total bill they would like to tip and then output the tip amount. Run your program to show the amounts on a bill of £87 with a tip of 12.5%. When outputting your total tip amount also print

Python Installation Videos

Muted excerpts of video walkthroughs demonstrating Python installation on 3 different operating systems.

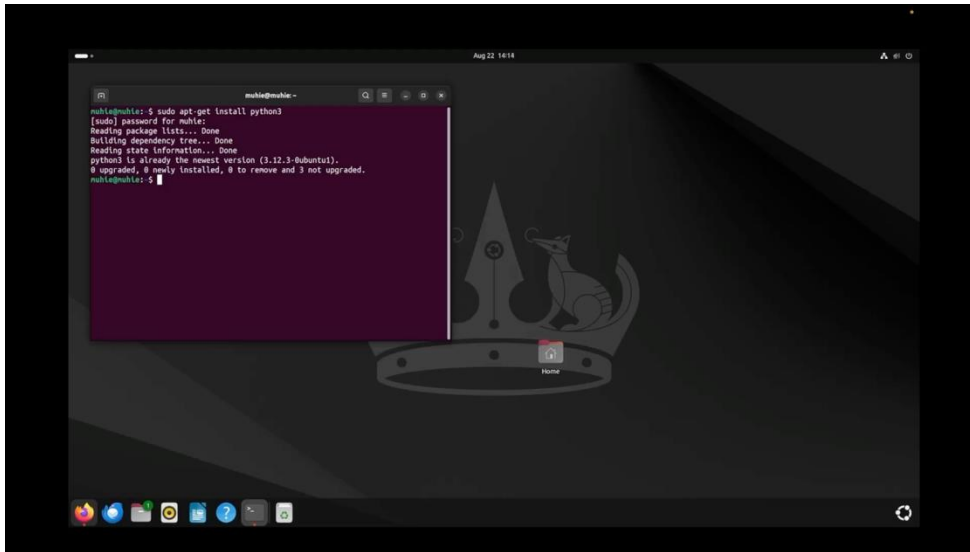


Figure 2: Python installation on Linux [2]

EMS412U How to install Python and an IDE on Mac(OS)

Muhie Al Haimus

Figure 3: Python installation on MacOS [2]



Figure 4: Python installation on Windows 11 [2]

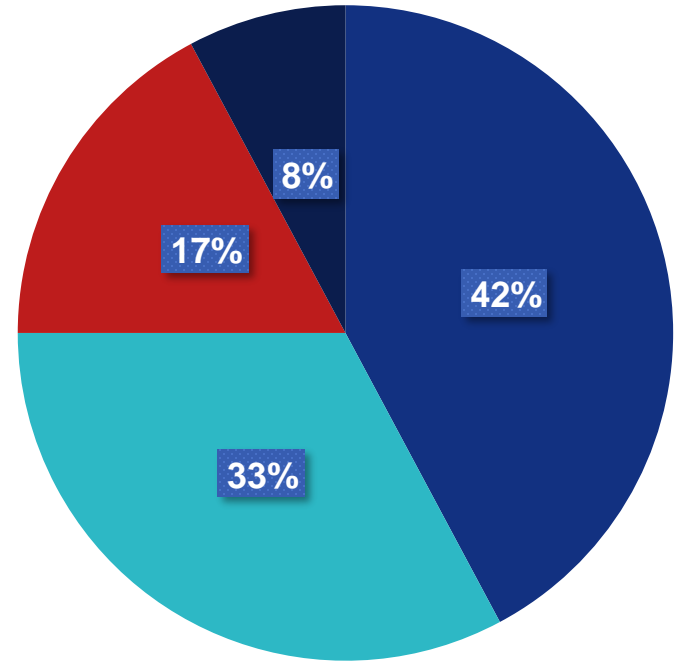
Insights from Feedback Survey

- **Ongoing**
- Preliminary results, based on **64 initial responses**
- **75%** of respondents indicated they had between **0 to 10 hours of programming experience.**

How would you describe your previous Python programming experience?

- Novice: no programming experience
- Beginner: 1-10 hours of programming experience
- Intermediate: 10-100 hours of programming experience
- Advanced: 100+ hours of programming experience

Figure 5: Question 1 of the feedback survey, where students rated their prior programming proficiency, with the number of hours indicated for each category



■ Novice ■ Beginner ■ Intermediate ■ Advanced

Figure 6: Pie chart illustrating the distribution of abilities among students who used the toolkit (sample size = 64)

Preliminary Student Feedback Survey Responses

What aspects of this toolkit did you find most helpful?

“The **practice questions are really useful**; they allowed me to understand python in a context outside of the JupyterHub environment”

“I like how **for each type of error, there's an explanation** (so you know why it is a common error and why it is incorrect) **and a solution** (so when you are stuck on your own work, you can look at this toolkit and correct it yourself, rather than asking for help). I like how there are **practice questions with solutions** at the end for you; allowing us to test our own knowledge and proving that the toolkit works.”

“The toolkit was very **organised**, and I felt that **I could independently go through the activities** without any problems.”

Preliminary Student Feedback Survey Responses

What suggestions, if any, do you have for improving this resource?

“Possibly, creating a more concise **toolkit with videos or explanations alongside questions**, to help people with no experience of coding get comfortable”

“Teaching **more python libraries** within the guides and **practice questions**”

“Create more **videos that go over programming exercises** in the toolkit”

Analysis of Interim Student Feedback

- Student feedback was **positive** with strong remarks to the **layout and presentation** of the information being delivered.
- Many students that were new to programming felt that the document provided a **smoother introduction** to the **concepts of python programming** and its syntax.
- However, many of the respondents **requested additional video tutorials** to accompany the toolkit, offering a more **in-depth, step-by-step teaching approach**. This suggestion can be considered for **future iterations**.
- Overall, there has been a **significant increase in positive feedback** regarding Python learning, with many students feeling **more confident** and **better prepared** for tackling Python questions in the EMS412U module.

Concluding Remarks and Next Steps

- Incorporate feedback and make necessary adjustments, including additional snippets and examples for future deployment
- Include more video/interactive resources to assist students that prefer different teaching styles
- Consider deployment for the second-year EMS506U module once the toolkit has been revised to emphasise data science and related topics
- Presenting at CREME (Centre for Research in Engineering and Materials Education), Festival of Education, Student engagement conference (Westminster)

References:

- 1) Coding is hard, www.youtube.com/@jomakaze
- 2) Python install playlist, https://www.youtube.com/watch?v=sRj58RCRRSE&list=PLVTKec-v1Xhsq0B_q3NbLEAafPIh_3XhE
- 3) SEMS Graduate Attributes, <https://www.qmul.ac.uk/queenmaryacademy/educators/resources/graduate-attributes/graduate-attributes-guidance-for-staff/>

Thank you for your time!



Queen Mary
University of London