Data Structures and Algorithms in Python

Muni Sreenivas Pydi

Data Science and AI Training for Trainers 2023/24

Course Outline

- Introduction to Python Programming (beginner)
- Introduction to Data Structures (beginner)
- Algorithm Analysis and Complexity (advanced)
- Sorting Algorithms (advanced)
- Searching Algorithms (advanced)
- Advanced Data Structures and Algorithms (advanced)
- Applications of Data Structures and Algorithms (beginner)

My definition of “advanced”: Slightly difficult content for someone new to programming/CS

Both beginner and advanced lectures will have some coding component.

Course Outline

- My definition of “advanced”: Slightly difficult content for someone new to programming/CS
- Both beginner and advanced lectures will have some coding component.
- The first three lectures build on each other. Rest of them are somewhat independent. Even if you skip one of the lectures, you may still be able to follow subsequent lectures.

Course Logistics

- Venue: PariSanté Campus
- Time: Every Monday 18h to 21h
- People
  - Main instructor: Muni Sreenivas Pydi muni.pydi@lamsade.dauphine.fr
  - Administration / Registration: Elise Bellier elise.bellier@psl.eu

Instructor Introduction: Muni

- Research Experience: AI Fellow, LAMSADE Laboratory, Dauphine-PSL, France [2022 -]
  - Research areas - Trustworthy machine learning (privacy, robustness, fairness)
- Teaching Experience:
  - Differential Privacy for Machine Learning, IASD Masters, PSL [Jan-Mar 2023, Jan-Mar 2024]
  - PSL AI Training for Trainers 2023/2024
    - Data Structures and Algorithms in Python [Feb-Apr 2024]
    - Data Management and SQL [Apr-Jun 2024]
- Education: PhD in Electrical & Computer Engineering
  - University of Wisconsin – Madison (UW-Madison), USA [2017 - 2022]
- Industry Experience: Software Engineer, Samsung R&D, India [2014 - 2016]
  - 4G/LTE protocol stack development

What to expect from this course

- An introduction to “Algorithmic thinking”
  - It teaches you how to look at a problem from an algorithmic perspective.
- Things you will learn:
  - Breaking down a problem into more manageable subproblems
  - Coming up with suitable abstractions / data structures to store your data efficiently
  - Identifying the right algorithms to solve your problem
  - Evaluating your solution for efficiency in terms of storage and running-time

What to expect from this course

- We will use Python extensively throughout the course
  - It is ok if you have not coded in Python before. It is very easy to pickup.
  - The tools / techniques we'll learn can be applied to any programming language, not just python.
  - The first lecture is devoted to a quick review of programming in python
- Every lecture will have a follow-along Google Colab notebook (a Jupyter notebook).
  - You are encouraged to bring your computing device to class to follow-along the exercises.
  - You will need a (free) google account to access the notebooks.

What NOT to expect from this course

- This course is not a programming course.
  - Corollary: this is not a python programming course.
  - It is not intended to teach you how to code.
  - It is not intended to teach you the syntax and all the features of any specific programming language.
- This course is not a “programming for data science” course.
  - This course does not teach you about NumPy / Pandas / Scikit-learn / Pytorch or other such popular machine learning / data science packages.

Why take this course

- In this course, you will learn many ways to sort a list of items (for example, sort a list of names in alphabetical order).
  - *”Why not simply learn the best way to sort and be done with it?”*
  - Best is subjective.
  - *“Python has built-in functions to sort pretty much anything. Why should I learn to implement it from scratch?” “I’m comfortable using the high-level libraries and frameworks for coding. Why should I learn low-level details like the implementation of data structures and algorithms?”*
  - It is always best to know the fundamentals. It helps you write efficient code, helps with debugging, and helps in choosing the right frameworks for your task.
  - *“I work in --- area. Why would I ever need to learn about this?”*
  - You may not know what will be relevant/useful for you until you see it.

Why take this course

- Data structures and algorithms are fundamental to computer science (CS).
- It trains your brain to think “algorithmically”.
- It allows you to take informed decisions, when it comes to applying CS tools in your own discipline.
- You will exposed to CS jargon like stacks/queues/time-complexity/O(n) etc. so you don’t have to feel out of place in meetings with CS people.
- It gives you necessary foundation to take more advanced CS courses.
- It exposes you to some of the coolest ideas in CS.
- It is fun, if you think solving puzzles is fun.

Resources

- Python ‘official’ documentation: https://docs.python.org/3/tutorial
- W3 Schools Python tutorial: https://www.w3schools.com/python/
- Practice problems: https://leetcode.com/, https://www.codechef.com/
- Free online textbook: https://runestone.academy/ns/books/published/pythonds/index.html
- Standard textbook: Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *”Introduction to algorithms”*. MIT press, 2022.

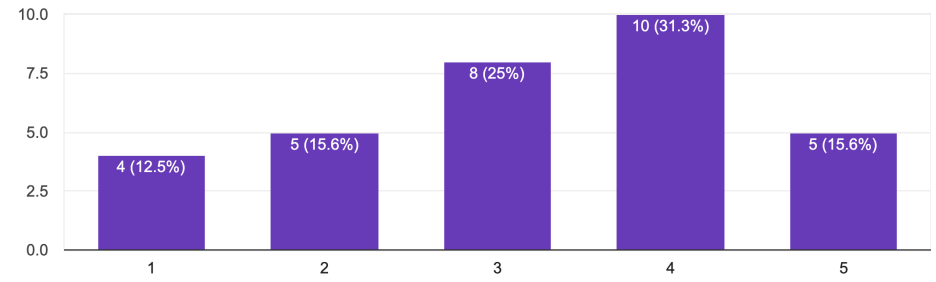
Intake Survey

- Link: https://forms.gle/mAw8d7f8Cw6UG6GF8
- QR code:

Intake Survey Results

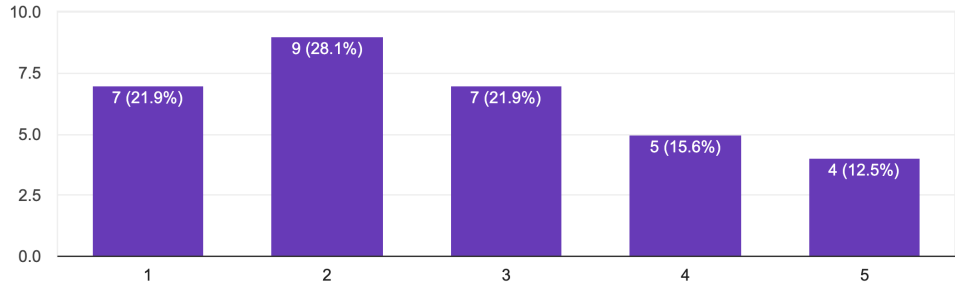
Please rate your exposure to programming

32 responses



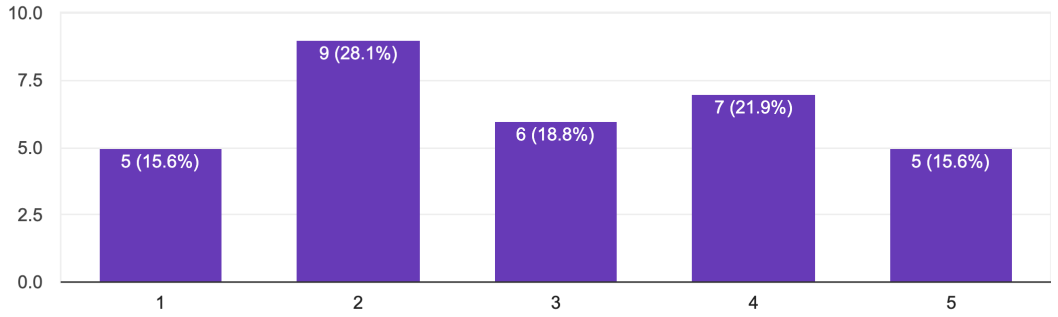
Please rate your exposure to Python

32 responses



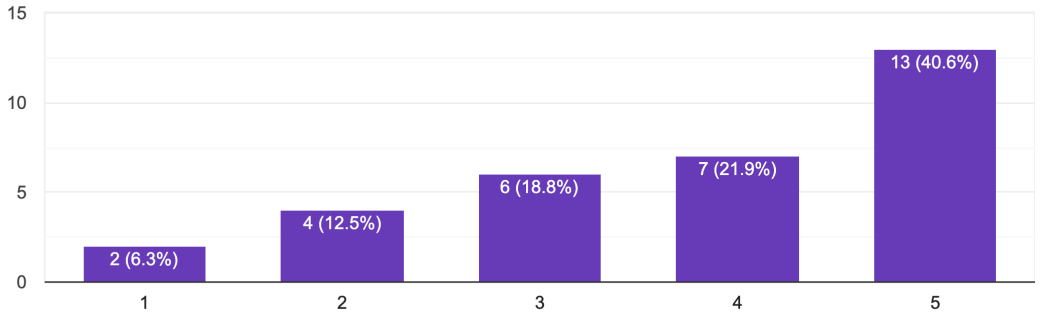
Please rate your exposure to data structures and algorithms

32 responses



How comfortable are you with mathematical formulas / equations

32 responses



Today’s Agenda

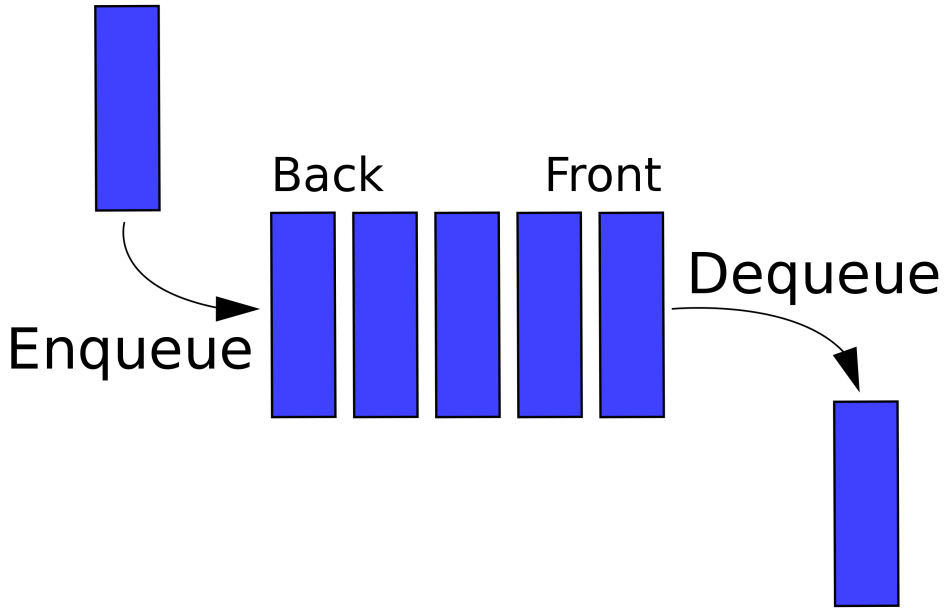
- A quick overview of programming in python
  - Colab notebook link
  - Please make a copy of the notebook in your own google drive. You can do this by going to ‘File’ -> ‘Save a copy to drive’. On your own copy, feel free to edit/experiment and play with the code as you wish!
- A quick overview and motivation for course contents

Data Structures

- A data structure is an abstract structure to organize, store and manage data.
- Example data structures:
  - Array
  - List
  - Stack
  - Queue
  - Tree
  - Hash map
  - Graph

Queue

- Queue is a sequence of items following a FIFO (First-In First-Out) principle.



Queue

- Examples of queues in real-world
  - Waiting for services at a bank
  - Waiting for your turn to be seated at a restaurant
  - Supermarket checkout
  - Traffic lanes at toll-booths
- In all these examples, new items are added at the end of the line and old items are taken out from the beginning of the line.

Stack

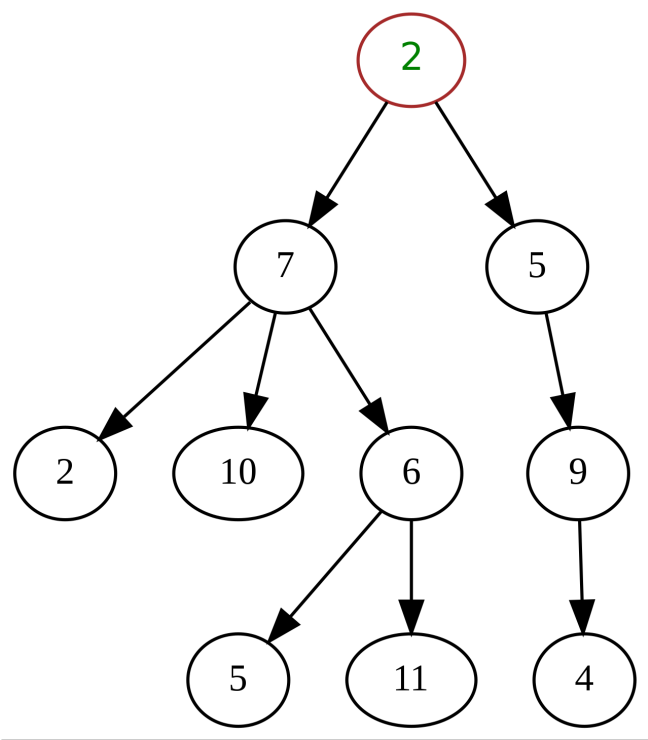
- Queue is a sequence of items following a LIFO (Last-In First-Out) principle.

Stack

- Examples of stacks in real-world
  - Web browser history when you press the back button
  - Undo functionality in software, for example: text editor
  - Plate/cup dispensers in cafeterias
  - Stack of books on your table
- In all these examples, items are added and removed only from one end of the line.

Tree

- Tree is a hierarchical data structure with a set of connected nodes



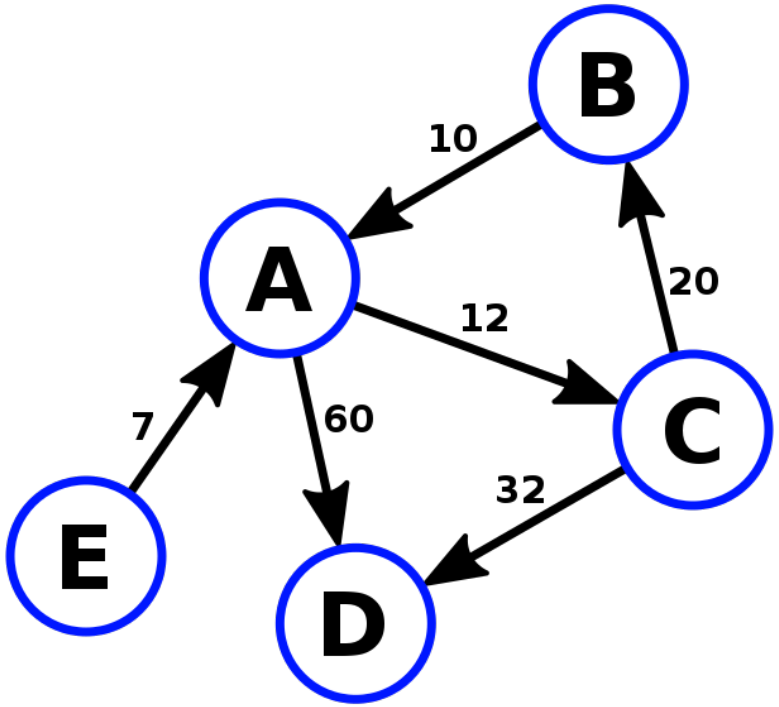
Tree

- Examples of trees in real-world
  - File system hierarchy
  - Family trees
  - Organizational charts
  - Biological taxonomy

- In all these examples, items are arranged in a hierarchical manner without cycles, as opposed to a simple list.

Graph

- Graph is a networked data structure with a set of connected nodes



Graph

- Examples of Graphs in real-world
  - Social networks
  - Transportation networks
  - Trading networks
  - The internet

- In all these examples, items are the vertices of a graph, some of which are connected by edges indicating specific relationships

Algorithms

- An algorithm is a sequence of instructions for solving a specific problem.
- Example algorithms:
  - Sorting algorithms
  - Searching algorithms
  - Machine learning algorithms
  - Cryptographic algorithms
  - Cooking recipes
  - Driving directions
  - Workout plans
  - Morning routines

Algorithm Examples

- Searching algorithms
  - How do you find a word in the dictionary? Linear search vs binary search
- Sorting algorithms
  - How do you sort a deck of cards? Insertion sort, bubble sort or merge sort
- Graph traversal
  - How do you find your destination metro station on a metro map? Breadth-first search or depth-first search