



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Higher Order Functions



## Announcements

---

- Labs:
  - Practice questions are required, but like the rest based on effort.
  - We messed up lab 1, but don't worry about it.
- Midterm: Oct 7, 7-9pm
  - We will be using Zoom to proctor, details in a couple weeks
  - Also we will have an alternate for conflicts
- My OH, normally Tues 12-1pm
- Everyone will get off the waitlist soon!
  - Dual enrolled in CS61A: We can optionally transfer early assignment scores.



## News: An AI “Publishes” and Op-Ed in the Guardian

---

- ...with help from a UC Berkeley student!
- [“A robot wrote this entire article. Are you scared yet, human?”](#)
- This article was written by GPT-3, OpenAI’s language generator. GPT-3 is a cutting edge language model that uses machine learning to produce human like text. It takes in a prompt, and attempts to complete it.
- The prompts were written by the Guardian, and fed to GPT-3 by Liam Porr, a computer science undergraduate student at UC Berkeley. GPT-3 produced eight different outputs, or essays. [...] we chose instead to pick the best parts of each, in order to capture the different styles and registers of the AI. Editing GPT-3’s op-ed was no different to editing a human op-ed. We cut lines and paragraphs, and rearranged the order of them in some places. Overall, it took less time to edit than many human op-eds.



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Higher Order Functions



## Learning Objectives

---

- Learn how to use and create higher order functions:
- Functions can be used as data
- **Functions can accept a function as an argument**
- Functions can return a new function



## Code is a Form of Data

---

- Numbers, Strings: All kinds of data
- Code is its own kind of data, too!
- Why?
  - More expressive programs, a new kind of abstraction.
  - ”Encapsulate” logic and data into neat packages.
- This will be one of the trickier concepts in CS88.



## What is a Higher Order Function?

---

- A function that takes in another function as an argument
- OR
- A function that returns a function as a result.



## An Interesting Example

---

$$\sum_{k=1}^5 k = 1 + 2 + 3 + 4 + 5 = 15$$

$$\sum_{k=1}^5 k^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225$$

$$\sum_{k=1}^5 \frac{8}{(4k-3) \cdot (4k-1)} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} = 3.04$$





UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Higher Order Functions



## Learning Objectives

---

- Learn how to use and create higher order functions:
- Functions can be used as data
- Functions can accept a function as an argument
- **Functions can return a new function**



## What is a Higher Order Function?

---

- A function that takes in another function as an argument
- OR
- A function that returns a function as a result.



UC Berkeley EECS  
Lecturer  
Michael Ball

# Computational Structures in Data Science

---



## Environments & Higher Order Functions



## Learning Objectives

---

- Learn how to use and create higher order functions:
- Functions can be used as data
- **Functions can accept a function as an argument**
- **Functions can return a new function**



## Example: compose

---

- Python Tutor:

```
http://pythontutor.com/composingprograms.html#code=def%20square%28x%29%3A%0A%20%20%20%20return%20x%20*%20x%0A%20%20%20%20%0As%20%3D%20square%0Ax%20%3D%20s%283%29%0A%0Adef%20make_adder%28n%29%3A%0A%20%20%20%20def%20adder%28k%29%3A%0A%20%20%20%20%20%20%20%20%20%2
```



## Environment Diagrams

---

- Organizational tools that help you understand code
- **Terminology:**
  - **Frame:** keeps track of variable-to-value bindings, each function call has a frame
  - **Global Frame:** global for short, the starting frame of all python programs, doesn't correspond to a specific function
  - **Parent Frame:** The frame of where a function is defined (default parent frame is global)
  - **Frame number:** What we use to keep track of frames,  $f_1$ ,  $f_2$ ,  $f_3$ , etc
  - **Variable vs Value:**  $x = 1$ .  $x$  is the **variable**, 1 is the **value**



## Environment Diagrams Steps

---

1. Draw the global frame
2. When evaluating assignments (lines with single equal), always evaluate right side first
3. When you call a function MAKE A NEW FRAME!
4. When assigning a primitive expression (number, boolean, string) write the value in the box
5. When assigning anything else, draw an arrow to the value
6. When calling a function, name the frame with the intrinsic name – the name of the function that variable points to
7. The parent frame of a function is the frame in which it was defined in (default parent frame is global)
8. If the value isn't in the current frame, search in the parent frame





## Environment Diagram Tips / Links

---

- NEVER EVER EVER draw an arrow from one variable to another.
- Source:
- [http://markmiyashita.com/cs61a/environment\\_diagrams/rules\\_of\\_environment\\_diagrams/](http://markmiyashita.com/cs61a/environment_diagrams/rules_of_environment_diagrams/)
- <http://albertwu.org/cs61a/notes/environments.html>