Updates and Announcements

- Midterm 2 Weeks!
- Oct 14, 7-9pm
- Room: 155 Dwinelle
- Will release samples soon
- HW Party: Tues 8-10pm, “Woz” (430 Soda)
  - Lab 4 and HW4 Help
  - Lab 5 and HW5 out then, to get a start!
- Python Tutor, use https://tutor.cs61a.org

Computational Concepts Toolbox

- Data type: values, literals, operations,
  - e.g., int, float, string
- Expressions, Call expression
- Variables
- Assignment Statement
- Sequences: tuple, list
  - indexing
- Data structures
- Tuple assignment
- Call Expressions
- Function Definition Statement
- Conditional Statement
- Iteration:
  - data-driven (list comprehension)
  - control-driven (for statement)
  - while statement
- Higher Order Functions
  - Functions as Values
  - Functions with functions as argument
  - Assignment of function values
- Lambda - function valued expressions
- Recursion
  - Next week!

Universality

- Everything that can be computed, can be computed with what you know now.
- Poorly or Well

Today’s Lecture

- Review
  - Higher Order Functions
  - Environments
  - Lambda
  - Some recursion + HOFs

What would Python Display?

```python
def summation(n, func):
    total = 0
    for i in range(1, n + 1):
        total = total + func(i)
    return total

def cube(x):
    return x*x*x

def sum_cubes(n):
    return summation(n, cube)
```

```
sum_cubes(3)
```

- A) 6
- B) 9
- C) 27
- D) 36
- E) An Error Occurs
Names can be Bound to Functional Arguments

Applying a user-defined function:

- Create a new frame
- Bind formal parameters \( (f \& x) \) to arguments
- Execute the body: \( \text{return } f(f(x)) \)

Lambda Expressions

- *Function expression*
  - “anonymous” function creation
  - Expression, not a statement, no return or any other statement

\[ \lambda \text{arg or arg_tuple} : \text{expression using args} \]

\begin{align*}
\text{add_one} & = \lambda v : v + 1 \\
\text{def add_one(v):} & \quad \text{return } v + 1
\end{align*}

Lambda Expressions

\begin{align*}
\text{inc_maker} & : \lambda x: x+i \\
\text{def inc_maker(i):} & \quad \text{return } \lambda x: x+i \\
\text{inc_maker(3)} & \quad \text{<function inc_maker.<locals>.<lambda> at 0x10073c510>} \\
\text{inc_maker(3)(4)} & \quad 7
\end{align*}

Lambda Expressions

\begin{align*}
\text{map(} & \lambda x: x \times x, \{1,2,3,4\} \\
\text{list(} & \text{map}(\lambda x: x \times x, \{1,2,3,4\})) \\
\text{[} & \text{1, 4, 9, 16}\text{]}
\end{align*}

What would Python Display?

- A) 5
- B) 7
- C) 8
- D) \text{<function } \lambda x: x \times x \text{> at 0xb859710}>
- E) An Error Occurs

Python Tutor Link

Demo

- Acronym
  - Filter
  - Map
  - Reduce
  - “The University of California at Berkeley” → ‘UCB’