Administrative Issues

• **Midterm exam: wed Oct 3 6-8 pm**
  - Room based on last digit of SID
  - 0-5 LeConte 1 (60%)
  - 6-9: VLSB 2040
  - Alternative and accommodations during 5-9 by request

• **Materials will go through 10/1 Lecture**

• **Please do mid-term survey**

• **Office hours start here after class and migrate down to BIDS in 190 Doe Library**

• **Live piazza thread 166**
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
• Recursion
• Lambda - function valued expressions
def qsort(s):
    """Sort a sequence - split it by the first element, sort both parts and put them back together."""

    if not s:
        return []
    else:
        pivot = first(s)
        lessor, more = split_fun(leq_maker(pivot), rest(s))
        return qsort(lessor) + [pivot] + qsort(more)

>>> qsort([3,3,1,4,5,4,3,2,1,17])
[1, 1, 2, 3, 3, 3, 4, 4, 5, 17]
Exploring Environments

```
def split(p, s):
    *** Returns a pair of lists based on applying predicate ***
    return [i for i in s if p(i)], [i for i in s if not p(i)]

def leq_maker(v):
    def leqv(x):
        return x <= v
    return leqv

def qsort(s):
    *** Sort a sequence by recursively splitting and sorting ***
    if not s:
        return []
    else:
        pivot = s[0]
        lessor, more = split(leq_maker(pivot), s[1:])
        return qsort(lessor) + [pivot] + qsort(more)
qsort([3,1,5,3,2,17])
```

[Diagram of Python 3.6 code and visualization of a function call tree]
lambda

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

\[
\text{lambda } \langle \text{arg or arg\_tuple}\rangle : \langle \text{expression using args}\rangle
\]

\[
\text{inc} = \lambda v : v + 1
\]

\[
\text{def inc(v):}
\]
\[
\quad \text{return } v + 1
\]
Lambda Examples

```python
>>> sort([1,2,3,4,5], lambda x: x)
[1, 2, 3, 4, 5]

>>> sort([1,2,3,4,5], lambda x: -x)
[5, 4, 3, 2, 1]

>>> sort(((2, "hi"), (1, "how"), (5, "goes"), (7, "I")),
      lambda x: x[0])
[(1, 'how'), (2, 'hi'), (5, 'goes'), (7, 'I')]

>>> sort(((2, "hi"), (1, "how"), (5, "goes"), (7, "I")),
      lambda x: x[1])
[(7, 'I'), (5, 'goes'), (2, 'hi'), (1, 'how')]

>>> sort(((2,"hi"),(1,"how"),(5,"goes"),(7,"I")),
      lambda x: len(x[1]))
[(7, 'I'), (2, 'hi'), (1, 'how'), (5, 'goes')]
```

http://cs88-website.github.io/assets/slides/adt/mersort.py
>>> def inc_maker(i):
...     return lambda x: x+i
...

>>> inc_maker(3)
<function inc_maker.<locals>..<lambda> at 0x10073c510>

>>> inc_maker(3)(4)
7

>>> map(lambda x: x*x, [1, 2, 3, 4])
<map object at 0x1020950b8>

>>> list(map(lambda x: x*x, [1, 2, 3, 4]))
[1, 4, 9, 16]

>>>
Thinking back over concepts

• Data type
  – Representation
    » literals and display
    » Internal representation
  – Set of operations
  – Conversions to other types

• Expressions – computation of values of a type
  – Built-in operations and function calls
  – Comprehensions

• Statements
  – Assignment & Control
  – Conditionals, Iteration

• Functions – objects and control