Higher-Order Functions
Announcements
Office Hours: You Should Go!

You are not alone!

https://cs61a.org/office-hours/
Iteration Example: Nice Numbers
Nice Numbers

Rounding off 2,799 to 2,800 makes it nice.

**Definition:** A nice number doesn't have 98, 99, 01, or 02 among its digits and 00 can only be followed by more 0's.

Not-so-nice numbers: 99 2,799 5,016 9,902 1,200,456 98,402,001

Nicer versions: 100 2,800 5,000 10,000 1,200,000 100,000,000

These numbers are nice enough already and unaffected: 755 2,859 45,622,895

Implement nice, which takes a positive integer n. It returns the nearest nice number to n.

• For numbers that end in 98 or 99 or 01 or 02, round to the nearest 100.
• Look for 98 or 99 or 01 or 02 among the digits that aren't at the end.

To solve a problem, describe a process and work through an example:

4 7 9 8 4 0 2 0 0 1

\[ \underline{4 7 9 8 4 0 2 0 0 1} \]

\[ \underline{4 7 9 8 4 0 2 0 0 0} \]

\[ \underline{4 7 9 8 4 0 0 0 0 0} \]

\[ \underline{4 7 9 8 4 0 0 0 0 0} \]

\[ \underline{4 8 0 0 0 0 0 0 0 0} \]

(Demo)
Designing Functions
Describing Functions

A function's *domain* is the set of all inputs it might possibly take as arguments.

A function's *range* is the set of output values it might possibly return.

A pure function's *behavior* is the relationship it creates between input and output.

```python
def square(x):
    """Return X * X.""
    x is a number
    square returns a non-negative real number
    square returns the square of x
```
A Guide to Designing Function

Give each function exactly one job, but make it apply to many related situations

```python
>>> round(1.23)       >>> round(1.23, 1)     >>> round(1.23, 0)       >>> round(1.23, 5)
  1               1.2             1               1.23
```

Don’t repeat yourself (DRY): Implement a process just once, but execute it many times

(Demo)
Higher-Order Functions
def cube(k):
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.\n    >>> summation(5, cube)
    225\n    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total

>>> summation(5, cube)
225

The cube function is passed as an argument value

0 + 1 + 8 + 27 + 64 + 125

The function bound to term gets called here
Program Design

Modularity
Abstraction
Separation of Concerns
Twenty-One Rules

Two players alternate turns, on which they can add 1, 2, or 3 to the current total

The total starts at 0

The game end whenever the total is 21 or more

The last player to add to the total loses

Some states are good; some are bad

\[ \text{21} \leftarrow \text{20} \leftarrow \text{18} \leftarrow \ldots \]

(Demo)
Functions as Return Values

(Demo)
Locally Defined Functions

Functions defined within other function bodies are bound to names in a local frame

```
def make_adder(n):
    """Return a function that takes one argument k and returns k + n."
    def adder(k):
        return k + n
    return adder
```

```
>>> add_three = make_adder(3)
>>> add_three(4)
7
```

A function that returns a function

The name add_three is bound to a function

A def statement within another def statement

Can refer to names in the enclosing function