Higher-Order Functions

Announcements

Office Hours: You Should Go!

You are not alone!

https://c88c.org/sp25/office-hours/

Control

While Statements

While statements contain statements that are repeated as long as some condition is true.

Important considerations:

- How many separate names are needed and what do they mean?
- The while condition **must eventually become a false value** for the statement to end (unless there is a return statement inside the while body).
- Once the while condition is evaluated, the entire body is executed.

Names and their initial values

```
i, total = 0, 0
while i < 3:
    i = i + 1
    total = total + i
```

The while condition is evaluated before each iteration

A name that appears in the while condition is changing

Executed even when is set to 3

Designing Functions

Describing Functions

```
def square(x):
    """Return X * X."""
```

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

x is a number

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

square returns a non-negative real number

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

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

```
>>> 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

Summation Example

```
def cube(k):
    return pow(k, 3)

def summation(n, term):
    """Sum the first n terms of a sequence.

    >>> summation(5, cube)
    225
    """
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total
```

Function of a single argument (*not called "term"*)

A formal parameter that will be bound to a function

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

(Demo)

Some states are good; some are bad

$$\begin{tikzcd}
 & & 19 \arrow[dl] & \\
21 & 20 \arrow[l] & 18 \arrow[l] & \cdots \\
 & & 17 \arrow[ul] &
\end{tikzcd}$$

(Demo)

Functions as Return Values

(Demo)

Locally Defined Functions

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

A function that returns a function

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

    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """
    def adder(k):
        return k + n
    return adder
```

The name add_three is bound to a function

A def statement within another def statement

Can refer to names in the enclosing function