More on functions

Announcements

- Lab 3 due tomorrow at midnight.
- Ninja session tonight, 7-9pm in this room.

Today's plan

- Review functions
- Stack diagrams

- Program for playing craps

Review

- A **function** is a named, parameterized block of code. Once we define and name a function, we can invoke or **call** it elsewhere in our program.
- A function call evaluates to the value **returned** by the function.

Review

```
def add5(n):
    return n + 5


def main():
    result = add5(10)
    # result is now 15
    print(result)


main()
```

Parameters and Arguments

- When we define a function we list its **parameters**.
- When we call a function we list its corresponding **arguments**.
- The arguments determine the initial values of the parameters, based on the order in which they are listed.
- Once initialized, parameters behave just like variables

Params and Args

```
def addTwice(n, m):
    n = n + 5
    # n is now 7
    return n + m

def main():
    result = addTwice(2, 10)
    # result is now 17
    print(result)

main()
```

Separate frames

- Each function gets its own **environment** or **frame** in which to define parameters and other variables.

- A variable in one frame is distinct from a variable in another frame, even if they share the same name.

- A variable can only be referred to in the function in which it was defined, after it has been initialized. This is sometimes called the variable's **scope**.

- We can use variables in one frame/function to initialize parameters in another function.

Separate frames

```
def add5(n):
    n = n + 5
    # add5's n is now 15
    return n

def main():
    n = 10
    result = add5(n)
    # result is now 15, main's n is still 10
    print(result)

main()
```

Control flow

- When a function is called, **control** passes from the calling function to the called function.

- Once the function has finished performing all the instructions in its definition, control passes back to the calling function at the place where it left off.

- Sometimes a function relinquishes control early with a **return** statement.

Control flow

```
def add5(n):
    return n + 5
    # This code is 'unreachable'
    print("I've already returned from add5")

def main():
    print(add5(10))

main()
```

Control flow

```
def sayHi(name):
    print("Now I'm in sayHi")
    print("Hi, %s" % name)
    print("I'm about to leave sayHi")


def main():
    user = raw_input("What's your name? ")

    print("\nI'm in main, about to call sayHi")
    sayHi(user)
    print("Now I'm back in main")

main()
```

Kinds of functions

- Some functions are called because of the return value they produce.
- Some functions are called because they perform side effects like printing.
- Some functions perform side effects and return something useful.
- A function's name is typically a verb indicating the task it performs.

Functions are useful

- Why use functions?
  - Keep our programs organized, manageable, easy to follow
  - For planning / outlining
  - Easier to maintain
  - Easier to reuse

Stack diagrams

- A dynamic visualization of the variables/parameters and their associated values in each active frame of a running program.

- pythontutor.com shows a version of the stack diagram, similar but not exactly the same as ours.

Stack diagrams

```
def add5(n):
    n = n + 5
    return n

def main():
    a = 10
    result = add5(a)
    print(result)

main()
```

craps.py