Tuples and Functions

Outline

Another Type of Value

Reusing Code with Functions

Quiz

Assignment 1

Values: Tuples

- One more value type for today
- Tuples are sequences of values
  - Each element can have any type
- We write them with parentheses and commas:
  - `(1, "hello", 3)` is a tuple
  - `("hey", "hi")` is another

Working with Tuples

- We can access the elements of a tuple using square braces:

  ```
  tup = (1, 2, "hi", 4)
  num_one = tup[0] # we start counting from 0
  hi_str = tup[2]
  ```

- There are many other things we can do with tuples
  - But that is a topic for another week

Pattern matching

We have one last trick for defining variables. What does the following code do?

```
tup = (1, 2, "hi", 4)
(num_one, num_two, hi_str, num_4) = tup
```

Experiments with Pattern Matching

What do the following code blocks do?

```
tup = (1, 2, "hi", 4)
(num_one, num_two, hi_str, num_4, num_5) = tup
```

```
tup = (1, 2, "hi", 4)
(num_one, num_two) = tup
```

```
(a, b, c) = ("x", "y", "z")
```

```
("a", "b", "c") = (x, y, z)
```

Functions

Maybe you've seen functions before in algebra or precalculus:

- $\mathrm{f}(x) = x + 2$
- $\mathrm{g}(x, y) = \sin(x) * \cos(y)$
- $\mathrm{h}(a) = f(a) + g(a, a)$

A function has a *name* and *parameters*.

Functions in Python

Python uses a notation similar to algebra for *calling* or *evaluating* functions:

```
import math
root = math.sqrt(2209)
num_str = str(root)
```

Functions may produce a *return value* (also called a *result* or *output*).

Import and packages (aside)

We often want to pull in additional Python code from other files.

Reusing other people's code is a superpower in programming.

Python comes with many useful packages:

- `import math` for math functions and constants
- `import datetime` for date and time operations
- `import random` to generate random numbers
- `import csv` to read comma-separated-value tables from files
- ... and many more!

Nested Names (aside)

- Earlier, we saw `math.sqrt(2209)`
  - The `.` in the middle of the name means:
  - "From the package `math`, get the variable named `sqrt`"
    - Which is a function!
- You may see this "nested name" syntax in the future
  - It's a lot like folders and files on your computer

Defining Functions

We can define functions in Python using the `def` keyword:

```
def hello():
    print("hello, world!")
```

`hello` uses the built-in `print` function, which outputs text to the terminal.

Functions and Procedures

What is the difference between these two functions?

```
def a(x):
    print(x)

def b(x):
    return x
```

Functions and Procedures

Python functions differ from algebraic functions in three main ways:

1. They may not return (*evaluate to*) anything at all
2. They might produce different outputs even if the input is the same
   - E.g., the output might depend on the time of day or a random number
3. They may not terminate, or may terminate with some kind of error

Exercise

What will the following Python program print?

```
def a(x, y):
    return x + y


def b(x):
    return a(x, x) + a(x, x)


print(b(5))
print(b("hi"))
print(b(a(0, 0)))
```

Multi-line functions

Python functions can span multiple lines. Python uses *indentation* to determine which lines are part of the function. What will this program output?

```
def c(x):
    y = x + x
    z = y + y
    return z + z
z = 5
print(z)
print(c(1))
print(z)
```

Scope

Last detail for today: variables and arguments *defined inside a function* are only available inside of that function.

```
def d(x):
    y = 10
    print(x + y)
print(x) # This is an error.
print(y) # So is this!
d(5) # even if we call d...
print(x) # This is still an error.
print(y) # So is this!
```

As we saw on the last slide, the z inside of `c` was *distinct from* the z outside of `c`.

Quiz

Learning Community

Be sure to meet with your LC today/tomorrow! You can work on the assignment in the same room, get rapid tech support, etc!