CSC110 Lecture 3: Comprehensions and Introduction to Functions

David Liu, Department of Computer Science

Navigation tip for web slides: press ***?*** *to see keyboard navigation controls.*

Introduction

Quick announcements

- Recruiting a volunteer note-taker! (Campuswire post #5)
- U of T AI club: LearnAI course sign-up! (Campuswire post #6)
- (Hart House Play) Truth Values: Exploring Gender, Diversity and Unconscious Bias in STEM (Campuswire post #7)
- Additional Resources module posted on Quercus
    - Student Success Strategies: Note-taking

The seven main Python data types

Data type	Description	Operations
`int`, `float`	Numeric data	Arithmetic (e.g. +), comparisons (e.g. ==, <)
`bool`	Boolean (True/False) data	`and`, `or`, `not`
`str`	Text data	==, +, `in`, indexing (`s[...]`)
`set`	Collection, no duplicates, no order	==, `in`
`list`	Collection, duplicates allowed, order matters	==, +, `in`, indexing (`s[...]`)
`dict`	Collection of association pairs	==, `in`, key lookup (`d[...]`)

Clarifying terminology

Every expression is a statement, but not every statement is an expression.

Every literal is an expression, but not every expression is a literal.

One “catch-up” point from yesterday

```
>>> {1, 'hi', True}
{1, 'hi'}
```

Key idea:

```
>>> True == 1
True
```

The Python interpreter treats `1` and `True` as duplicates in a set.

Learning objectives

In this lecture, you will learn to:

1. Create collections in Python using comprehensions.
2. Create sequences of integers in Python using `range`.
3. Define terminology relating to functions in mathematics and programming.
4. Name and describe some built-in Python functions.
5. Recognize and write Python code for function call expressions.
6. Recognize and write Python code for function definitions.

Comprehensions

In mathematics, we use set builder notation to express large (possibly infinite!) sets:

$$\{x^2 \mid x \in \mathbb{N}\} = \{0, 1, 4, 9, \ldots\}$$

“The set of x^2 values where x ranges over the natural numbers.”

In Python, we can use set comprehensions to express sets.

```
>>> nums = {0, 1, 2, 3, 4, 5}
```

```
>>> {x ** 2 for x in nums}
{0, 1, 4, 9, 16, 25}
```

Set builder notation

$$\{x^2 \mid x \in \mathbb{N}\}$$

Set comprehension expression

```
{x ** 2 for x in nums}
```

Two other comprehension types

List comprehension:

```
>>> nums = {0, 1, 2, 3, 4, 5}
>>> [x ** 2 for x in nums]
[0, 1, 4, 9, 16, 25]
```

Dictionary comprehension:

```
>>> nums = {0, 1, 2, 3, 4, 5}
>>> {x : x ** 2 for x in nums}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

General comprehension syntax

Set comprehension:

```
{ <expression> for <variable> in <collection> }
```

List comprehension:

```
[ <expression> for <variable> in <collection> ]
```

Dictionary comprehension:

```
{ <key_expr>: <value_expr> for <variable> in <collection> }
```

Design process for comprehensions

> ***Problem**: Given the set `numbers = {1, 2, 3, 4, 5}`, compute a new set containing the reciprocals ($\frac{1}{\square}$) of each number.*

1. Identify the type of comprehension to use.

   - `set`

2. Start with the “identity comprehension” of this type.

```
>>> {x for x in numbers}
```

3. Modify the left subexpression to compute the desired result.

```
>>> {1 / x for x in numbers}
```

Exercise 1: Practice with comprehensions

https://www.teach.cs.toronto.edu/~csc110y/fall/lectures/03-comprehensions-and-built-in-functions/worksheet/

`range`: a sequence of numbers

For integers `m` and `n`, `range(m, n)` represents the sequence of numbers `m`, `m + 1`, ..., `n - 1`.

Note: the start of range is inclusive, but the end of the range is exclusive. This ensures the size of `range(m, n)` is always `n - m`.

`range` in comprehensions

> ***Problem**: compute the reciprocals of the numbers between 1 and 20, inclusive.*

Demo!

Exercise 2: Comprehensions and `range`

Comprehensions with multiple variables

Consider this new set operation, the **Cartesian product**:

$$A \times B = \{(x, y) \mid x \in A \text{ and } y \in B\}$$

Example:

$$\{1, 2\} \times \{10, 20\} = \{(1, 10), (1, 20), (2, 10), (2, 20)\}$$

We can do this in Python as well: **demo!**

Functions in Python

Code we’ve seen so far:

- literals (`3`, `'hello'`, `[1, 2, 3]`)
- operators (`+`, `-`, `and`)
- variables and assignment statements (`numbers = {1, 2, 3}`)
- comprehension expressions (`{x ** x for x in numbers}`)

How do we build up code with these elements to perform useful computations?

Recall a mathematical definition of a **function**: a mapping of elements from one set A (called the function's domain) to a set B (called the function's codomain). Notation:

$$function : A \to B$$

Example:

$$f : \mathbb{R} \to \mathbb{R}$$
$$f(x) = x^2$$

Functions take in inputs and return outputs.

- $f(5) = 25$
- $f(0) = 0$
- $f(-1.5) = 2.25$

Functions in Python

In Python, functions do the same thing: take in input values and return an output value.

But Python functions aren’t just limited to numbers!

Demo: some built-in Python functions

- `abs`
- `len`
- `sum`
- `sorted`
- `max`/`min`
- `type`
- `help`

Terminology

```
>>> abs(-5)
5
```

- `abs(-5)` is a function call expression
- `abs` is the name of the function being called
- `-5` is an argument
  - or, “`-5` is passed to `abs`”
- `abs(-5)` returns `5`
  - `abs(-5)` evaluates to `5`

Exercise 3: Practice with built-in functions

Defining our own Python functions

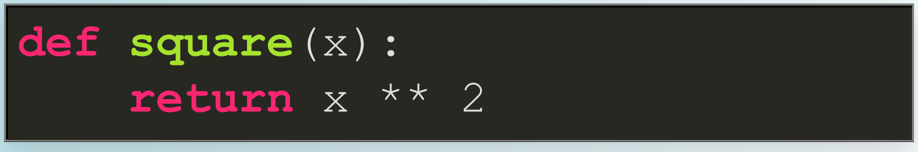
We can define our own mathematical functions just by writing them down:

$$
\begin{gathered}
f:\mathbb{R}\to\mathbb{R}\\
f(x)=x^2
\end{gathered}
$$

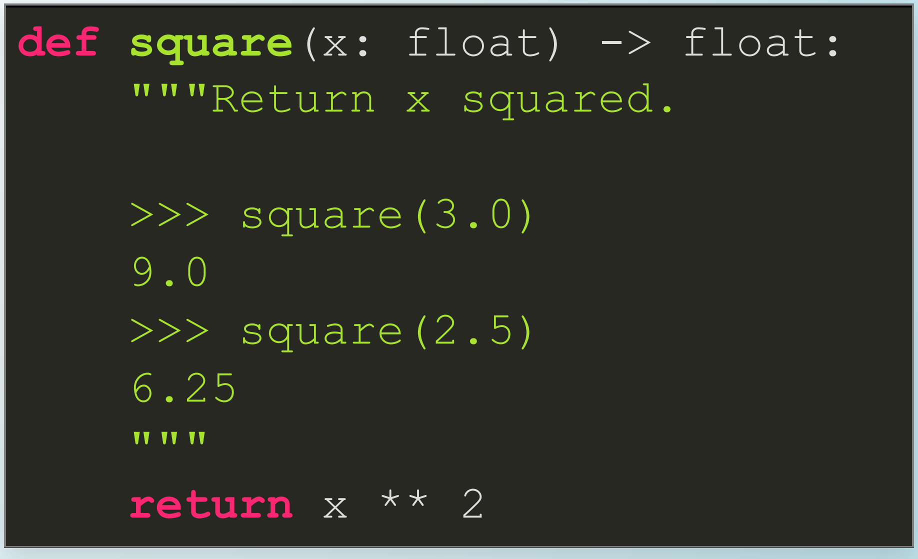
How do we define our own functions in the Python programming language?

$$f : \mathbb{R} \to \mathbb{R}$$

$$f(x) = x^2$$



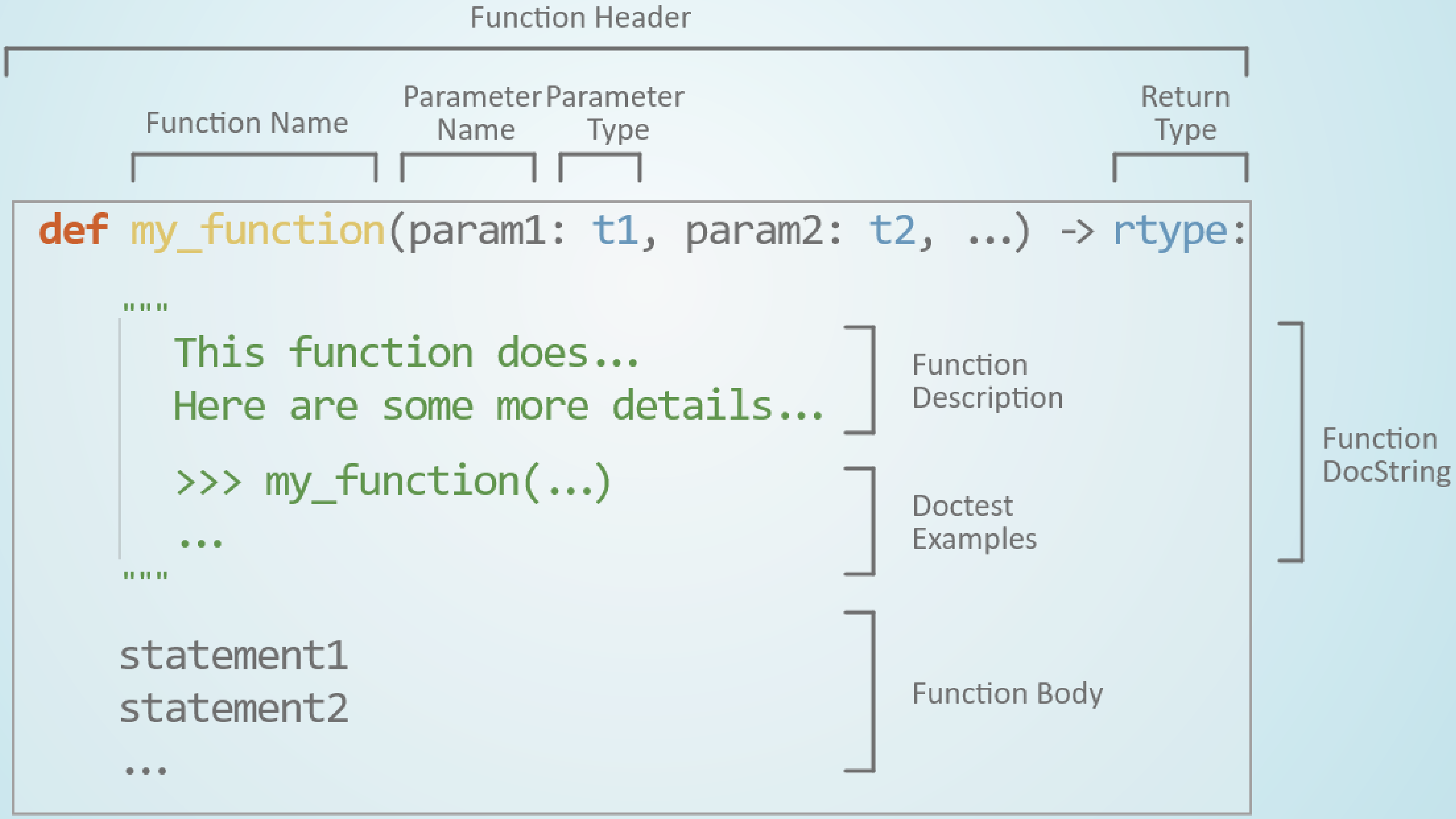
```python
def square(x):
    return x ** 2
```



```python
def square(x: float) -> float:
    """Return x squared.

    >>> square(3.0)
    9.0
    >>> square(2.5)
    6.25
    """
    return x ** 2
```

Anatomy of a function definition



Function Header

Function Name

Parameter Name

Parameter Type

Return Type

```
def my_function(param1: t1, param2: t2, ...) -> rtype:

    """
      This function does...
      Here are some more details...

      >>> my_function(...)
      ...
    """

    statement1
    statement2
    ...
```

Function Description

Doctest Examples

Function DocString

Function Body

Demo: writing code in a Python file

Summary

Today you learned to...

In this lecture, you learned to:

1. Create collections in Python using comprehensions.
2. Create sequences of integers in Python using `range`.
3. Define terminology relating to functions in mathematics and programming.
4. Name and describe some built-in Python functions.
5. Recognize and write Python code for function call expressions.
6. Recognize and write Python code for function definitions.

Homework

- Readings from today: 1.7, 2.1, 2.2
- Reading ahead:
  - Thursday: 2.4, 2.7
  - Tutorial 1: 1.8
  - Next Monday: 2.3, 2.5, 2.6, 2.8
- Prep 2 and Assignment 1 will be posted tomorrow!

That feeling when you reach the end of a lecture and see a meme: