

CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding
Sep 20 2019, Lecture 11



Announcement

- Home assignment 2 posted, due next Friday midnight.
- Lab 2 will be posted this weekend.
- Quiz next Friday (Sep 27)
 - may take ~10mins at the beginning of the class
- Midterm Exam will be on Oct 18 (Friday, after fall break).

Agenda

- Agenda:
 - Review

Python Indentation

- Indentation refers to the spaces at the beginning of a code line
- Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important
- Python uses indentation to indicate a block of code
 - if statements, while/for loops, functions, class, etc.

Python Comments

- Comments can be used to explain Python code, make the code more readable.
- Comments start with a `#`, and Python will ignore everything after `#`
- No multiple line comments like other languages, use multiple `#` instead in Python
 - Example of bad comment

```
b = 59 # assign the value of 59 to b
```

- Example of good comment

```
salestax10 = 1.10 # defining a sales tax 10%  
salestax20 = 1.20 # defining a sales tax 20%
```

Python Variables

- Variables are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it

For example, Create two variables to store age and name

```
age = 5  
name = "xinyi"  
  
print(age)  
print(name)
```

Python Variables

- Variable Names
 - Variable names can contain only letters, numbers, and underscores. They can start with a letter or an underscore, but not with a number.
 - Spaces are not allowed in variable names, but underscores can be used
 - Avoid using Python keywords and function names as variable names
 - Variable names should be short but descriptive.
 - Case Sensitive

For example, first_name, message_1, age are valid names, but not 1_message, print

Python Data Types

- In programming, data type is an important concept
- Variables can store data of different types, and different types can do different things.
- Python has many built-in data types, most often used include:
 - Text Type: `str`
 - Numeric Type: `int`, `float`
 - Sequence Type: `list`, `tuple`, `str`
 - Mapping Type: `dict`
 - Boolean Type: `bool`

Python Numbers

- There are three numeric types in Python:

- int

- $x = 4$

- $y = 123456$

- float

- $\pi = 3.1416926$

- $e = 2.71828$

- complex

Operator	meaning	Examples
+	plus - Add two operands	$x+y$
-	Minus - subtract right operand from the left	$x-y$
*	Multiplication- multiply two operands	$x*y$
/	Division - divide left operand by the right one	x/y
%	Modulus - remainder of the division of left operand by the right	$x\%y$
//	Floor division - division that results into whole number adjusted to the left in the number line	$x//y$
**	Exponent - left operand raised to the power of right	$x**y$

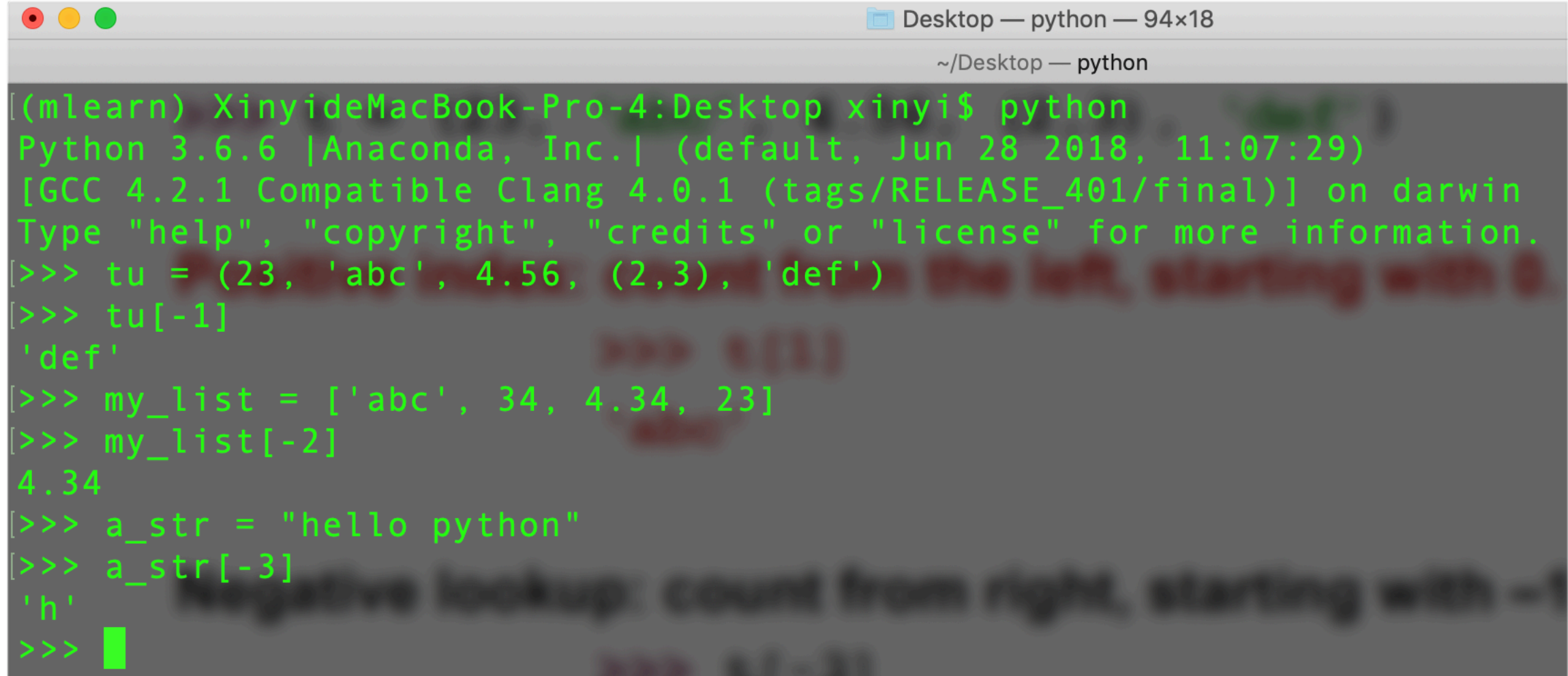
Arithmetic operators

Python Sequence Type

- List
 - **Mutable** ordered sequence of items of **mixed types**
- Tuple
 - A simple **immutable** ordered sequence of items
 - Immutable: a tuple cannot be modified once created....
 - Items can be of **mixed types**, including collection types
- String
 - **Immutable**
 - Conceptually very much like a tuple

Python Indexing

- Positive index: count from the left, **starting with 0**
- Negative lookup: count from the right, **starting with -1**

A screenshot of a macOS terminal window titled "Desktop — python — 94x18". The window shows a Python 3.6.6 prompt. The user enters a tuple 'tu' and a list 'my_list', then uses negative indexing to access elements from the right. The output shows 'def' for the last element of the tuple and '4.34' for the second-to-last element of the list. The user then enters a string 'a_str' and uses negative indexing to access the first character 'h'.

```
(mlearn) XinyideMacBook-Pro-4:Desktop xinyi$ python
Python 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:07:29)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> tu = (23, 'abc', 4.56, (2,3), 'def')
[>>> tu[-1]
'def'
[>>> my_list = ['abc', 34, 4.34, 23]
[>>> my_list[-2]
4.34
[>>> a_str = "hello python"
[>>> a_str[-3]
'h'
[>>> ]
```

Changing, Adding, and Removing Elements of A List

- Modifying elements in a List
 - To change an element, use the name of the list followed by the index of the element you want to change, and then provide the new value you want that item to have
- `append()`
- `pop()`, `del`

Python Strings

- String literals in python are surrounded by either single quotation marks, or double quotation marks.
- You can use three quotes for multiline string
- String methods you will encounter most often:
 - `.lower()`
 - `.strip()`
 - `.split()`
 - `.format()`
 - ...

Python Dictionaries

- Dictionaries store a **mapping** between a set of keys and a set of values
 - Keys can be any **immutable** type
 - Values can be any type
 - Values and keys can be of different types in a single dictionary
- You can define/add/modify/delete key-value pairs of a dictionary

```
>>> credentials = {'alice': 'this is a password', 'carl': 'this is also a password'}
>>> print(credentials['carl'])
this is also a password
>>>
```

Conditional test (Boolean expression)



- Programming often involves examining a set of conditions (conditional test) and deciding which action to take based on those conditions.
- **If** statement allows you to examine the current state of a program and respond appropriately to that state
- Return Boolean data type: **True** or **False**

Conditional test (Boolean expression)

- Check for Equality
 - Note: single = for assignment, == for check equality
- Check for Inequality
 - use !=
- Numerical comparison
- Combine boolean expressions using logical operators
 - and, or

Python if/else

- if statements syntax

indentation (4 spaces)  *if conditional_test:* **colon** 
do something
then do something

- Be careful when using whitespace for indentation
 - use the same number of spaces for indentation. PEP-8 recommends 4 whitespaces. (2 spaces, tab are also valid)
 - you can use any number of spaces in other cases, though valid, but not recommended. (say, `x = 5`)

Python if/else

- Use if-elif chain vs Use multiple simple if statements

```
1 requested_toppings = ["mushrooms", "extra cheese"]
2
3 if "mushrooms" in requested_toppings:
4     print("Adding mushrooms.")
5
6 elif "pepperoni" in requested_toppings:
7     print("Adding pepperoni.")
8
9 elif "extra cheese" in requested_toppings:
10    print("Adding extra cheese.")
11
12 print("Finished making your pizza!")
13
```

elif "pepperoni" in requested_t...

if_statements x

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/if_statements.py
Adding mushrooms.
Finished making your pizza!
```



```
1 requested_toppings = ["mushrooms", "extra cheese"]
2
3 if "mushrooms" in requested_toppings:
4     print("Adding mushrooms.")
5
6 if "pepperoni" in requested_toppings:
7     print("Adding pepperoni.")
8
9 if "extra cheese" in requested_toppings:
10    print("Adding extra cheese.")
11
12 print("Finished making your pizza!")
13
```

if_statements x

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week2/if_statements.py
Adding mushrooms.
Adding extra cheese.
Finished making your pizza!
```

Python while loop

- while loop syntax

indentation (4 spaces)  *while conditional_test:* **colon** 
 do something
 then do something

- It will keep execute the code block as long as the conditional test is true.
- usually you will need to modify the the values used in the conditional test once some conditions are met

Python for loop

- For-each is Python's **only** form of for loop, this is less like the **for** keyword in other programming languages.
- A **for** loop steps through each of the items in a collection type (list, dictionary, etc) or any other type of object which is "iterable" (remember when we call .keys() method of a dictionary)
- Often used with lists and dictionaries

indentation (4 spaces)



```
for <each item> in <collection>:  
    <statements>
```

colon

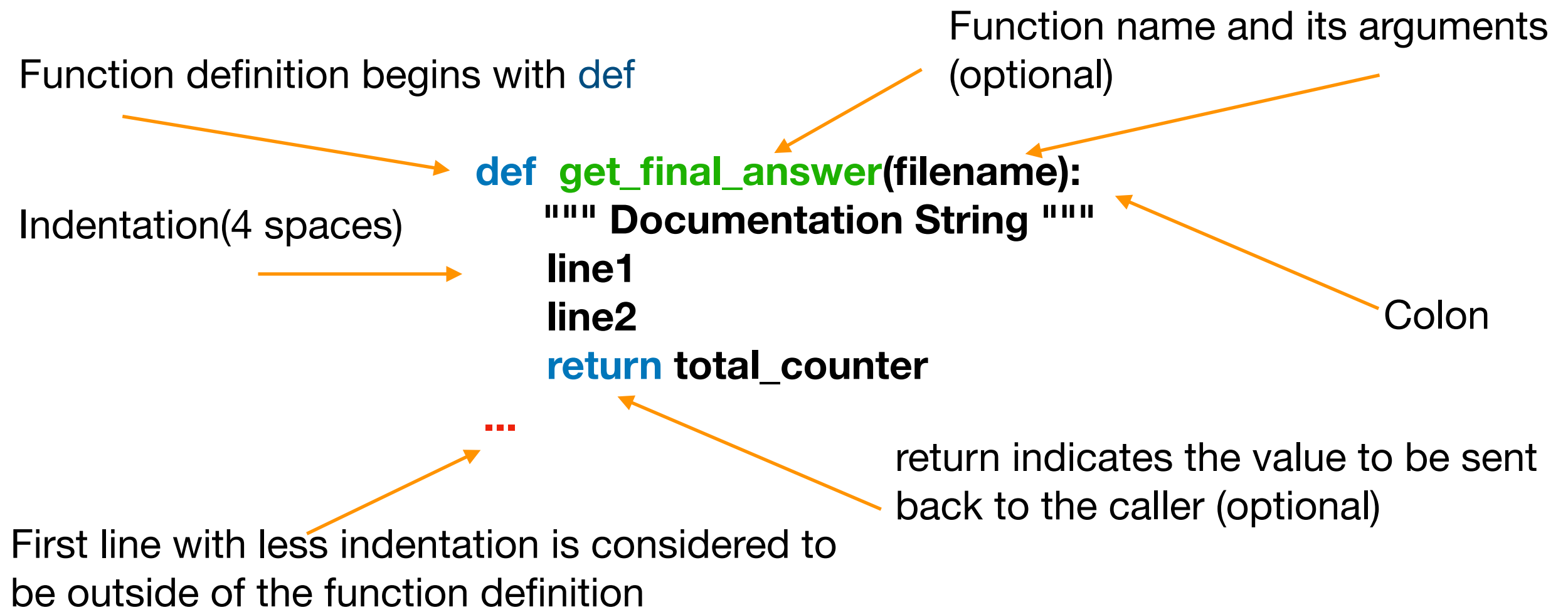


Python while/for loops

- Using **break** to exit a loop
 - To exit a loop immediately without running any remaining code in the loop
- Using **continue** in a loop
 - Rather than breaking out of a loop entirely without executing the rest of its code, you can use the **continue** statement to return to the beginning of the loop based on the result of a conditional test

Python Functions

- A function
 - A block of code which only runs when it is called
 - One way to organize and reuse code
 - You can pass information to a function
 - You can ask a function to return data



Python Functions

- Passing arguments
 - A function definition can have multiple parameters, a function call may need multiple arguments
- Ways of passing arguments
 - positional arguments
 - need to be in the same order the parameters were written
 - keyword arguments
 - where each argument consists of a variable name and a value

Python Input/Output

- Up till now, our program were static. The value of variables were defined or hard coded into the source code.
- To Allow flexibility we might want to take the input from the user. We use `input()` function in python

`input([prompt])`

Where **prompt** is the string we wish to display on the screen, it is optional

Python File Handling

- The key function for working with files is the `open()` function.
- The `open()` function takes two parameters: *filename* and *mode*
- There are four different modes for opening a file
 - `"r"` - Read-Default value. Opens a file for reading, error if the file does not exist
 - `"a"` - Append- Opens a file for appending, creates the file if it does not exist
 - `"w"` - Write- Opens a file for writing, create the file if it does not exist
 - `"x"` - Create -Creates the specified file, returns an error if the file exists

Python File Handling

- With Statement
 - The keyword **with** closes the file once access to it is no longer needed
 - **with** syntax:

Indentation(4 spaces) → **with open(filename) as the_file:**
 contents = the_file.read()
 print(contents) ← Colon

...
First line with less indentation is considered to be outside of the with code block