

Functions and Python

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 02, Spring 2021

Friday February 12

Project notes

- **Project 1 is due next Tuesday at 11:59pm.** Remember to submit your Google Doc report on Google Classroom and Lab 1 and Project 1 code on in one zip file (see lab/project page for details). Name your zip file `username.zip` (e.g. `owlayton.zip`).
- Writing is an important part of this class, please take the written reports seriously (but be concise and to the point, you don't need to write a novel!).
- Keep an eye on the rubric on Google Classroom! It is how your project will be graded.
- **Keep backups of your code** (e.g. Dropbox, Google Drive, etc)! It's very frustrating to lose your hard work!
- [Take a look at extensions.](#)

Academic integrity

- There will be zero tolerance for violation of academic integrity.
- Feel free to discuss projects with your classmates at a high/conceptual level (e.g. "I'm designing a house by drawing a square on top of a triangle").
- **As soon as code is involved, be careful!**
 - It is ok to share names of commands that are *generally* useful (e.g. " Hey, did you know there is a useful backward turtle command?").
 - It is NOT ok to show classmates your code files or send them your code.
 - I read your code and reports carefully; if I believe there is strong evidence of academic dishonesty, I will need to file a report.

Nobody wins if you copy code

- *You don't learn:* If you encounter a similar problem in the future, you won't know how to approach it.
- *False advertising:* You are misleading your future employer. You can't do the job you are hired to do.
- *In science:* your results don't mean anything; science doesn't advance by copying.
- *Working on a product:* What you develop isn't actually innovative. You don't fully understand what you have because you didn't create it.

Functions

Functions in Python

Functions make it easy to automate and do a sequence of actions whenever we want.

It allows us to define the set of operations that solve a sub-task of a larger problem.

Breaking down our morning routine:

```
def morningRoutine():  
    getOutOfBed()  
    brushTeeth()  
    getDressed()  
    eatBreakfast()
```

Running the morning routine on any day

The function **defines** what a morning routine is, but we actually need to **call** (or execute/run) it to do the routine on a given day:

```
def morningRoutine():  
    getOutOfBed()  
    brushTeeth()  
    getDressed()  
    eatBreakfast()
```

```
morningRoutine()
```

- The **parentheses ()** tell Python that we want to call/run the function: `morningRoutine()`
- Calling the function is **outdented** from the morning routine commands (not part of routine).

Example: Draw rectangle in Python

Adding a parameter

```
def morningRoutine(foodChoice):  
    getOutOfBed()  
    brushTeeth()  
    getDressed()  
    eatBreakfast(foodChoice)
```

- The **parameter** `foodChoice` allows us to pick what we want to eat on a given day for breakfast
 - Monday: `morningRoutine('eggs')`
 - Tuesday: `morningRoutine('cereal')`

Syntax: "Sentence punctuation" for code

- Parameters and function must always be one word. There are 2 main naming conventions:
 - Camel case: `foodChoice` (1st word starts lowercase)
 - Underscore: `food_choice` (everything lowercase)
- Commands belonging to a function must be indented.
- There is only one space between `def` and function name:

```
def morningRoutine():
```

- There must be a colon after the function definition parentheses `()`.

Multiple parameters

```
def morningRoutine(foodChoice, shirtColor):  
    getOutOfBed()  
    brushTeeth()  
    getDressed(shirtColor)  
    eatBreakfast(foodChoice)
```

...and we can call our function:

```
morningRoutine('toast', 'blue')
```

Example: Use parameters to control
rectangle's side lengths

Commenting out code 1/3

If a line of code starts with `#` symbol, Python will ignore/skip over it.

```
getOutOfBed()  
# I better find time to brush later!  
# brushTeeth()
```

- Useful for code **readability**: allows you to annotate/leave notes to help other humans understand what your lines of codes mean/are doing.
- Helpful for quickly changing which function is called when you run a code file.

Commenting out code 2/3

Draw a triangle when running your code file:

```
triangle()  
# rectangle()
```

Draw a rectangle when running your code file:

```
# triangle()  
rectangle()
```

Commenting out code 3/3

Easily toggle whether a line of code is commented out in VS Code:
Cmd-/

Example: Switching between
drawing triangle and our rectangle

Debugging tips: fixing common
issues for why your code won't run

Can't find your file

You get an error that looks like: `[Errno 2] No such file or directory` in Terminal.

- **What it means:** The file isn't in the current directory that Terminal is in.
- **How to fix it:** Use `pwd` to figure out where you are. Use `ls` to see all files in the current directory. If you aren't in the right folder, `cd` into the right one (remember folder icon drag trick).
- Files/folders in Terminal may be **case-sensitive** (*lower vs upper case letters in file and folder names matter*).

Syntax errors (1/2)

You forgot some code "punctuation" so Python doesn't know how to run your code.

What's missing here?

```
def triangle()  
    forward(100)  
    right(120)  
    forward(100)  
    right(120)  
    forward(100)
```

- The colon in the function definition!

Syntax errors (2/2)

What's missing here?

```
def triangle():  
    forward(100)  
    right(120)  
    forward(100)  
    right(120)  
    forward(100)
```

triangle

- The parentheses () when calling triangle!

Typos

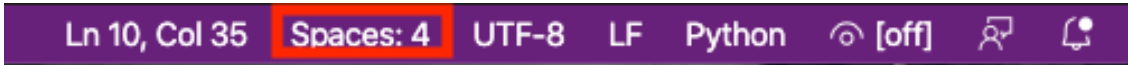
Typos are bad in English, but worse in programming because the computer is very literal and can't "figure out what you mean". What's the typo here that leads to an error?

```
def triangle():  
    forward(100)  
    right(120)  
    forward(100)  
    right(120)  
    forward(100)
```

```
trinagle()
```

- When calling triangle, the function is misspelled as trinagle!

Tabbing/White space error

- In Python, HOW lines of code are indented affects how Python runs your code.
- Even though spaces (from spacebar key) and tabs (from tab key) LOOK the same, they are different things. Python needs EITHER tabs (from tab key) or spaces (from spacebar), but DOES NOT like a mix of both.
- Changing default in VS Code: Click this 

Click Indent Using Spaces then click 4.

- Let's look at a few examples of mistakes caused by indentation/white space.