

Symbol lookup tables, coding style, operators and data types

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 04, Fall 2020

Wednesday September 2

Homework worksheets and quiz

- Short homework "worksheet" on Google Classroom.
- Due before class on Friday (**10am EST**). Hard deadline because I post solutions on Google Classroom.
- Graded: 1 or 0. 1 means you made a serious effort.
- Goal of worksheet: prepare you for the quiz.
- Quiz is 15 mins, similar to homework worksheet, [take on Moodle](#) closed book.
- Usually, I will give you homework worksheet on Mondays. Due date on Thursday noon (12pm EST).
- On some weeks, we will start on the the worksheet in-class.

Symbol lookup tables

Keeping `turtle` functions separate

- From now on, keep functions from `turtle` module separate from your own.
- Before: `from turtle import *`
- Now: `import turtle`

More comments on comments

Let's write `lecture_04_hexagon_style.py`

Summary: proper code organization and style

1. Header comments (2 sets of triple quotes). Include filename, your name, course, semester, one line description of what the file does.
2. Import statements
3. Any function definitions, separated by another two empty lines (each should have indented commands right below the function **signature** (`def hexagon(sideLength):`)). **Every function should have a docstring (""" style comment).**
4. At least one inline comment per function (`#` style), if your function has 3 or more lines of code.
5. A comment indicating where 'main' code is going to appear (either `' ' '` or `#` style).
6. Main code you want your program to execute.

Code organization and style guidelines

There are guidelines on how Python code should be formatted to improve readability and establish consistency across the appearance of different peoples' code. You can read more about the recommendations here: [PEP8](#).

Fun tip: as keyword to name your turtle

- Running turtle functions with `import turtle` like usual:

```
turtle.right(90)
```

```
turtle.forward(100)
```

```
turtle.left(45)
```

- Same output running turtle functions, giving turtle another name:

```
import turtle as byte
```

```
byte.right(90)
```

```
byte.forward(100)
```

```
byte.left(45)
```

Python interpreter: Operators, data types and printing