

Modules, Symbol tables, Code style

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 04, Spring 2021

Wednesday September 12

Follow-up

- Remainder ("mod") % operator.
- Finish operators: Type conversion, information loss, and rounding.

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 talk about best practices in code organization and style: revisit drawing a hexagon with `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

- Optionally, instead of `import turtle` you can give your turtle a name. Let's call him 'byte'.
- Running `turtle` functions with `import turtle` (no name):

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

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

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

- Same output running `turtle` functions with `import turtle as byte` (named):

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

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

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