LECTURE 3

PYTHON TOUR PART II

DOCSTRINGS, FUNCTIONS, MODULES, CLASSES

MCS 275 Spring 2021
Emily Dumas

LECTURE 3: PYTHON TOUR II

Course bulletins:

- Read the syllabus completely, ask if unclear.
- Discord open (link in the zoom chat or Blackboard).
- No class on Monday (MLK holiday)
- Quiz 1 will be posted at Noon on Tuesday (Jan 19)

NOTES FOR SELF STUDY

I expanded the written Python tour to include today's material:

Python tour (prep for MCS 275)

If you find something confusing or unfamiliar there, let me know and also check the textbook or the relevant MCS 260 slides.

DOCSTRINGS

The first non-comment statement in any Python file should be a string on a line by itself. That string should describe the file. It is called a **docstring**.

Docstrings can also appear as the first statement inside a function body or class definiton.

Anywhere else you want to put explanatory text, use a comment.

FUNCTIONS

Named*, reusable code blocks you can call (run) from elsewhere in your code.

```
def divisble_by_7(x):
    """Return True if x is divisible by 7"""  # <-- docstring!
    return x % 7 == 0

# ... and then later ...

if divisible_by_7(10987654321):
    print("Hey, did you know 10987654321 is a multiple of 7?!")
```

*It is also possible to define unnamed (anonymous) functions using `lambda`, but that isn't discussed in this quick overview.

See *Lutz*, Chapters 16-18 or MCS 260 Lec 9 and Lec 15.

MODULES

A module keeps a bunch of related code in one place; good for reuse and organization. The statement

```
import modulename
```

will look for `modulename.py` in current directory, or a built-in module with that name, and make its functions, classes, etc. available.

Use `modulename.funcname(...)` to call a function in a module.

See *Lutz*, Chapters 22-23 or MCS 260 Lec 20.

CLASSES

Classes let you define custom types in Python with **attributes** (data) and **methods** (behavior).

```
class Point:
    """A point in the xy-plane"""  # <--- Remember docstring!
    def __init__(self,x,y):
        """Initialize new point instance"""
        self.x = x    # make a new attribute (self.x)
        self.y = y    # make a new attribute (self.y)
    def translate(self,dx,dy):
        """Move the point by a vector (dx,dy)"""
        self.x += dx
        self.y += dy

P = Point(1,2)  # calls __init__(...)
P.translate(5,0)
print("After moving, P.x is",P.x)  # will print 6
```

See *Lutz*, Chapters 27-28 and MCS 260 Lec 23.

REFERENCES

- The Python tour is an expanded version of the live coding examples from today's lecture.
- Individual slides refer to chapters from *Lutz* (Learning Python 5ed).
  - Free access to online book for UIC students; see course web page.
- The MCS 260 Fall 2020 home page has slide presentations, sample code, and other resources for review.

REVISION HISTORY

- 2021-01-14 Initial publication