Python Wrap-Up

Springboard

About Python, Redux

Python Is …

- **high-level**: you think at a relatively high-level
- **dynamic**: running script can create its own functions/classes
- **dynamically-typed**: same variable can be used for int/string/etc
- **strongly-typed**: "a" + 3 doesn't eval to "a3"
- **compiled**

Python Is Compiled

```
def add(x, y, double=False):
    # do the adding
    result = x + y
    return result * 2 if double else result
```

gets "compiled" into "bytecode":

```
4           0 LOAD_FAST                0 (x)
            2 LOAD_FAST                1 (y)
            4 BINARY_ADD
            6 STORE_FAST               3 (result)

6           8 LOAD_FAST                2 (double)
           10 POP_JUMP_IF_FALSE       20
           12 LOAD_FAST                3 (result)
           14 LOAD_CONST               1 (2)
           16 BINARY_MULTIPLY
           18 RETURN_VALUE
      >>   20 LOAD_FAST                3 (result)
           22 RETURN_VALUE
```

You don't do this compilation separately.

It happens when you first run/import Python file.

Previously-compiled version is stored in ***__pycache__/add.pyc***

You don't need those file in Git — they get created when needed

Python Can Have Type Hints

```
def add(x: int, y: int) -> int:
    """Add x and y and return results."""

    return num1 + num2
```

- Editors can use this to help find errors
- Can produce prettier help/API documentation

Python Can Be Lazy

this works great…

```
def find_liked_num(nums):
    """Prompt user until they like a number."""

    for num in nums:
        if input(f"Do you like {num}? ") == 'y':
            return num
```

works great for this…

```
find_liked_num([1, 3, 4, 8])
```

If we wanted to do that for "all even numbers" …

```
find_liked_num([2, 4, 6, 8, ...])
```

Laziness Through *yield*

we can do this …

```
def evens(start):
    """Yield even numbers starting at start."""

    while True:
        yield start
        start = start + 2
```

then we can do this…

```
find_liked_num(evens(start=8))
```

yield is like "return this value now, and remember where it left off"

Laziness Is Good

It's nice to be able to loop over data …

- even if it's infinite (like all even numbers)
- or it's just too huge to hold in memory
- or it's expensive to pre-calculate when you might only need some

A lot of big-data stuff relies on this

There are even lazy list comprehensions: *generator expressions*

Operator Overloading

In both JS and Python, some operators (like `+`) mean different things, depending on the types of objects being acted on:

JavaScript

```
3 + 5    // 8

"hello " + "Whiskey"    // "hello Whiskey"
```

Python

```
3 + 5    # 8

"hello " + "Whiskey"    # "hello Whiskey"
```

In Python, you can "overload" an operator in a custom class: that operator can mean something different, and you can control that

Case-Insensitive Strings

demo/cistr.py

```
class CIString(str):
    """Subclass of string that is case-insensitive.

        >>> CIString("apple") == CIString("Apple")
        True

        >>> CIString("apple") < CIString("Banana")
        True
    """

    def __eq__(self, other):
        "Is self == other?"
        return self.lower() == other.lower()

    def __lt__(self, other):
        "Is self < other?"
        return self.lower() < other.lower()

    def __le__(self, other):
        "Is self <= other?"
        return self.lower() <= other.lower()
```

Python Libraries

Python Standard Library

Lots of useful data structures and features:

- queues and stacks
- binary search trees
- statistics
- complex numbers, fractions, cool math stuff
- functional programming helpers

Beautiful Soup

A lot of sites have APIs that return data.

Many don't, and you need to "scrape" HTML to get data.

Beautiful Soup is a terrific library for this.

Common Data Science Libraries

Numpy
: Super-fast linear algebra and matrix math

Pandas
: Data slicing/grouping/querying

SciKit-Learn
: Common machine learning algorithms

Good place to start

Jupyter

Jupyter is "interactive computing"

- Like IPython in a web page
- Can mix in documentation, drawings, code snippets
- Often used to play with data or share analyses
- Can publish on the web
- Can even interactively edit as a group!

And it's not just for Python :)

Zen Of Python

```
Beautiful is better than ugly
Readability counts

Explicit is better than implicit

Simple is better than complex
Complex is better than complicated

Special cases aren't special enough to break the rules
Although practicality beats purity

Errors should never pass silently

In the face of ambiguity, refuse the temptation to guess

If the implementation is hard to explain, it's a bad idea
If the implementation is easy to explain, it may be a good idea
```