October 2016 meeting.

PyRVA Sub-lightning talk

Today's topic: an example of a list comprehension.

slides: github.com/georgeflanagin/pyrva

First time only: my bio

George Flanagin
(me@georgeflanagin.com)

- work at UR as a computer scientist, and taught computer science at VCU
- have been working in Python 3 daily (and nightly) for two years,
- have a background in compiler writing, approximate pattern matching, and natural language processing.

For noobs: what is a list comprehension?

A list comprehension replaces subscripting, loops, and maps with compact one-liners.

Wrong way to build "x"

```
x = []
for _ in range(0, len(b)):
    x.append(foo(b[_]))
```

Python way to build "x"

```
x = [ foo(_) for _ in b ]
```

The Problem with Example List Comprehensions

- Some list comprehension examples are trivial
- Some list comprehension examples are synthetic
- Some list comprehension examples are … incomprehensible.

Maybe this one is better.

Today's Example

We needed a work-day calendar.
These are usually called "holiday calendars"

Given a date (often today), we need to know if today is a workday, or …
… what is the next workday, or …
… what was the most recent workday.

Simple concept for holiday calendar

(Build a list of either all the exceptions)
XOR
(Build a list of all the work-days)

AND Find out if the date you want is in (or not in) the list.

What we decided to do

- Make a UR Julian calendar.
- Fill it with the days from say t-10 days to t+400 days (rarely do we know any schedule more than one year into the future)
- Be positive, and test for inclusion.

Step 1: UR Julian Calendar

University of Richmond was founded 1 August 1830. That's Day Zero for us.

```
UR_ZERO_DAY = datetime.datetime(1830, 8, 1)
def urdate(dt:datetime.datetime = None) -> int:
    """
    Return number of days since 1 August 1830.
    """
    if dt is None: dt = datetime.datetime.today()
    return (dt - UR_ZERO_DAY).days
```

Step 2: Define the calendar

```
urcalendar['bizdays'] = [1,2,3,4,5]

urcalendar['holidays'] = [
    "November 24 2016",
    "December 25 2016", "January 1 2017",
    "January 16 2017", "May 29 2017",
    "July 4 2017", "September 4 2017"
    ]
```

Step 3: The concept of a workday...

The Python test needs to be simple and clear

```
isWorkday = (d in urcalendar.bizdays and
             d not in urcalendar.holidays)
```

Step 4: Redefine search

This is really what we want:

```
isWorkday = d in biglistofdays
```

But:

- Searching a long list looks like we don't know what we are doing.
- Someone is always going to whine "What about efficiency???"
- How do we build this list of days in a clear manner?

Step 5: The fix for inefficiencies

Naturally, there are batteries-included Python modules to help us.

```
import dateutil
import sortedcontainers
```

- dateutil gives us the ability to parse user-readable strings into datetime objects.
- sortedcontainers gives us the ability to binary search a long list in O(log N) time.

Step 6: Let's transform the holidays from text to urdate-s

```
urcal['holidays'] = [
    "November 24 2016",
    "December 25 2016", "January 1 2017",
    "January 16 2017", "May 29 2017",
    "July 4 2017", "September 4 2017"
    ]
             .... becomes ....

urcal['holidays'] = [ urdate(dateutil.parser.parse(_))
                            for _ in urcal['holidays']]
```

Comprehensions are read right to left.

```
urcal['holidays'] = [ urdate(dateutil.parser.parse(_))
                              for _ in urcal['holidays']]
```

Translation:

- for _ in urcal['holidays'] .... look at each text string in the holidays list
- dateutil.parser.parse(_) ... Parse it!
- urdate(...) ... Change it into an integer offset from 1 August 1830
- [..] ... make a new list
- change the reference of urcal['holidays'] to the new list.

Step 7: Let's bite off the bigger one

```
workdays = [ _ for _ in
    range(start-10, start+400)
      if _ % 7 in urcal['bizdays']
         and _ not in urcal['holidays']]
```

- range(start-10, start+400)   .... from ten days ago to 400 days from now.
- if _ % 7 in urcal['bizdays'] ... is it a weekday?
- and _ not in urcal['holidays'] ... is it NOT a holiday?
- workdays = [_ for _ in ..] ... make a new list

Step 8: Let's look at the calendar code as a whole.

```
def biglistofdays(urcal:dict) -> sortedcontainers.SortedList:
    """ """
    start = urdate()
    urcal['holidays'] = [ urdate(dateutil.parser.parse(_))
        for _ in urcal['holidays']]
    return sortedcontainers.SortedList([
        _ for _ in
        range(start-10, start+400)
        if _ % 7 in urcal['bizdays']
        and _ not in urcal['holidays']])
```

Step 9: Too much of a good thing?

```
def biglistofdays(urcal:dict) -> sortedcontainers.SortedList:
    """ """
    return sortedcontainers.SortedList(
        [ d for d in
            range(urdate()-10, urdate()+400)
            if d % 7 in [1,2,3,4,5]
            and d not in
            [ urdate(dateutil.parser.parse(_))
                for _ in urcal['holidays']]
        ])
```