Announcements

- Homework
  - Homework 7 is posted!
  - I'm a bit behind on homework grading, but probably will not catch up until the weekend
- Midterm a week from Friday!
  - We'll talk more end of the week about preparation and format
  - I do have an old test and some study questions I can give you
- Polling: `rembold-class.ddns.net`

Review Question

The below snippet of code is run. What would this expression evaluate to?

```
['One', 2, True][-1:1:-1][0]
```

A) `['One']`
B) `2`
C) `True`
D) None of the above, or this will error

Solution: `True`

Do you Comprehend?

- We will frequently use `for` loops and list appends to construct lists
- Nothing wrong in doing it the way we've done in the past
- Python does offer a more compact and nice way to combine these sorts of actions though
  - List Comprehensions!

Constructing a List Comprehension

- Still need the individual parts:
  - []'s for the list
  - for variable in sequence
- What was:

```
L = []
for x in range(10):
    L.append(x**2)
```

- Can become:

```
L = [x**2 for x in range(10)]
```

List Comprehensions further explained

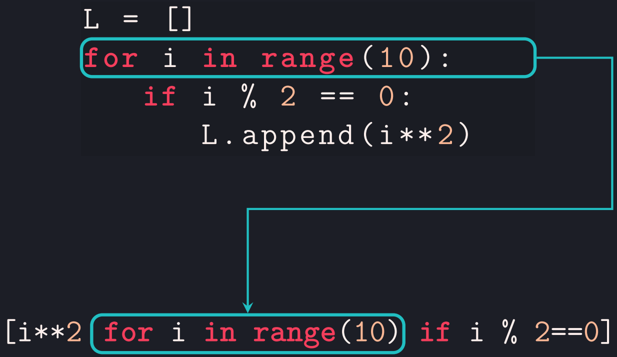
- Combines `for` loops and `list`s
- If you don't want both, a list comprehension is probably not what you want

```
L = []
for i in range(10):
    if i % 2 == 0:
        L.append(i**2)
```

```
[i**2 for i in range(10) if i % 2==0]
```

List Comprehensions further explained

- Combines `for` loops and `lists`
- If you don't want both, a list comprehension is probably not what you want



```
L = []
for i in range(10):
    if i % 2 == 0:
        L.append(i**2)
```

```
[i**2 for i in range(10) if i % 2==0]
```

List Comprehensions further explained

- Combines `for` loops and `list`s
- If you don't want both, a list comprehension is probably not what you want

```
L = []
for i in range(10):
    if i % 2 == 0:
        L.append(i**2)
```

```
[i**2 for i in range(10) if i % 2==0]
```

List Comprehensions further explained

- Combines `for` loops and `list`s
- If you don't want both, a list comprehension is probably not what you want

```python
L = []
for i in range(10):
    if i % 2 == 0:
        L.append(i**2)
```

```python
[i**2 for i in range(10) if i % 2==0]
```

Functional Objects

- Functions are first-class objects!
  - Can treat them just like any other type of object (`int`, `list`, etc)
- Can appear in
  - expressions
  - arguments to other functions
  - elements of lists
  - etc

Follow the `map`

- The builtin `map` function is like a more general purpose `apply_func_2_list`
- Simplest just takes a single argument function and a list:

```
def g(x):
    return 4*x + 2

xs = [1,2,3,4,5,6]
for result in map(g,xs):
    print(result)
```

Complex Mappings

- Can take multiple argument functions if given multiple lists

```
def h(x,y):
    return 4*x + 2*y

xs = [1,2,3,4,5,6]
ys = xs[::-1]
hs = [res for res in map(h,xs,ys)]
print(hs)
```