Functions as Objects

- Functions are **first class objects:**
  - They have types
  - They can be elements of data structures like lists
  - They can appear in expressions
    - As part of an assignment statement
    - As an argument to a function!!
- Particular useful to use functions as arguments when coupled with lists
  - Aka **higher order programming**

Example

```
def applyToEach(L, f):
    """assumes L is a list, f a function
       mutates L by replacing each element,
       e, of L by f(e)"""
    for i in range(len(L)):
        L[i] = f(L[i])
```

Example

```
def applyToEach(L, f):
    for i in
  range(len(L)):
        L[i] = f(L[

applyToEach(L, abs)
print(L)

applyToEach(L, int)
print(L)

applyToEach(L, fact)
print(L)

applyToEach(L, fib)
print(L)
```

```
L = [1, -2, 3.4]
```

Example

```
def applyToEach(L, f):
    for i in
  range(len(L)):
        L[i] = f(L[i])

applyToEach(L, abs)
print(L)

applyToEach(L, int)
print(L)

applyToEach(L, fact)
print(L)

applyToEach(L, fib)
print(L)
```

```
L = [1, -2, 3.4]
```

```
[1, 2,
  3.3999999999999]
```

Example

```
def applyToEach(L, f):
    for i in
  range(len(L)):
        L[i] = f(L[i])

applyToEach(L, abs)
print(L)

applyToEach(L, int)
print(L)

applyToEach(L, fact)
print(L)

applyToEach(L, fib)
print(L)
```

```
L = [1, -2, 3.4]

[1, 2,
  3.3999999999999]

[1, 2, 3]
```

Example

```
def applyToEach(L, f):
    for i in
  range(len(L)):
        L[i] = f(L[i])

applyToEach(L, abs)
print(L)

applyToEach(L, int)
print(L)

applyToEach(L, fact)
print(L)

applyToEach(L, fib)
print(L)
```

```
L = [1, -2, 3.4]

[1, 2,
  3.3999999999999]

[1, 2, 3]

[1, 2, 6]
```

Example

```
def applyToEach(L, f):
    for i in
  range(len(L)):
        L[i] = f(L[i])

applyToEach(L, abs)
print(L)

applyToEach(L, int)
print(L)

applyToEach(L, fact)
print(L)

applyToEach(L, fib)
print(L)
```

```
L = [1, -2, 3.4]

[1, 2,
  3.3999999999999]

[1, 2, 3]

[1, 2, 6]

[1, 2, 13]
```

Lists of functions

```
def applyFuns(L, x):
    for f in L:
        print(f(x))

applyFuns([abs, int, fact, fib], 4)
4
4
24
5
```

Generalizations of higher order functions

- Python provides a general purpose HOP, `map`
- Simple form – a unary function and a collection of suitable arguments
  - `map(abs, [1, -2, 3, -4])`
  - `[1, 2, 3, 4]`
- General form – an n-ary function and n collections of arguments
  - L1 = [1, 28, 36]
  - L2 = [2, 57, 9]
  - map(min, L1, L2)
  - [1, 28, 9]