Week 5b

Map, Filter, Reduce

Map and Filter

map() and filter()

```
L = [9, 2, 1, 3, 4, 5, 6]

>>> map(lambda x: x > 2, L)
<map object at 0x...>

>>> list(filter(lambda x:x>2,L))
[9, 3, 4, 5, 6]
```

map() and filter()

```
L = [9, 2, 1, 3, 4, 5, 6]

>>> tuple(map(lambda x: ‘o’ if x%2 else ‘e’,L))
(‘o’, ‘e’, ‘o’, ‘o’, ‘e’, ‘o’, ‘e’)

>>> list(filter(lambda x: ‘o’ if x%2 else ‘e’,L))
[9, 2, 1, 3, 4, 5, 6]
```

map() and filter()

```
L = [9, 2, 1, 3, 4, 5, 6]

>>> list(map(str,list(filter(lambda x:x%2,L))))
['9', '1', '3', '5']

>>> str(list(filter(lambda x:x>30,map(lambda x:x*x,L))))
'[81, 36]'
```

Exercise: Scale/Square Tuples

In Lecture

- We have talked about how to scale a list

```
[5,1,4,9,11,22,12,55]
```

```
[10,2,8,18,22,44,24,110]
```

- Or square a list

```
[5,1,4,9,11,22,12,55]
```

```
[25, 1, 16, 81, 121, 484, 144, 3025]
```

Tuple Scaling

List

```
def seqScaleI(seq,n):
    output = []
    for i in seq:
        output.append(i*n)
    return output


def seqScaleR(seq,n):
    if not seq:
        return seq
    return [seq[0]*n]+seqScaleR(seq[1:],n)
```

Tuple

- Try for 5 min
- No need to code from scratch, copy the list version and modify

Tuple Scaling

List

- What needed to be changed?

```
def seqScaleI(seq,n):
    output = []
    for i in seq:
        output.append(i*n)
    return output

def seqScaleR(seq,n):
    if not seq:
        return seq
    return [seq[0]*n]+seqScaleR(seq[1:],n)
```

Tuple

```
def seqScaleIT(seq,n):
    output = ()
    for i in seq:
        output += (n*i,)
    return output

def seqScaleRT(seq,n):
    if not seq:
        return seq
    return (seq[0]*n,)+seqScaleRT(seq[1:],n)
```

Tuple Map (our version)

```
L = [9, 2, 1, 3, 4, 5, 6]
```

```
>>> map(lambda x:x*x, L)
[81, 4, 1, 9, 16, 25, 36]

>>> map(lambda x:x*2, L)
[18, 4, 2, 6, 8, 10, 12]
```

```
def map(f,seq):
    output = []
    for i in seq:
        output.append(f(i))
    return output
```

Tuple Map (our version)

List

```
def map_il(f, seq):
    output = []
    for i in seq:
        output.append(f(i))
    return output

def map_rl(f, seq):
    if not seq:
        return seq
    return [f(seq[0])] + map_rl(f, seq[1:])
```

Tuple

```
def map_it(f, seq):
    output = ()
    for i in seq:
        output += (f(i),)
    return output

def map_rt(f, seq):
    if not seq:
        return seq
    return (f(seq[0]),) + map_rt(f, seq[1:])
```

Challenge!

Write an integrated function for **BOTH** list and tuples in **ONE** single function

Exercise: Sum Digits Square

Sum Digits

Write a function `sds(n)` which returns the sum of every digit in n.

```
>>> sds(123456789)
45
>>> sds(1111111111)
10
>>> sds(100000000000)
1
```

```
def map(f,seq):
    output = []
    for i in seq:
        output.append(f(i))
    return output
```

- Use our version of **map()** + built-in function **sum()**
  - But map() only applies to lists?
  - Hint: Given an integer N, what is `list(str(N))` ?

```
>>> list(str(123456))
['1', '2', '3', '4', '5', '6']
```

Sum Digits

```
>>> list(str(123456))
['1', '2', '3', '4', '5', '6']
>>> map(lambda x:x*x, list(str(123456)))
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    map(lambda x:x*x, list(str(123456)))
  File "G:/My Drive/Courses/CS1010E/CS1010E TA Folders/Tutor
k 08 map filter fold reduce/Wk08 more about sequence Tutoria
ap
    output.append(f(i))
  File "<pyshell#25>", line 1, in <lambda>
    map(lambda x:x*x, list(str(123456)))
TypeError: can't multiply sequence by non-int of type 'str'
>>> def sds(n):
        return sum(map(lambda x:int(x),list(str(n))))
```

Sum Digits Square

Write a function `sdss(n)` which returns the sum of the square of every digit in n.

```
>>> sdss(22222)
20
```

- 5 min

```
>>> def sds(n):
        return sum(map(lambda x:int(x),list(str(n))))
```

int(x) ** 2

Exercise: Taylor Series

Trigonometric Functions

- Can we use our map() for Taylor series?

Use higher order functions for this?

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \quad \text{for all } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \quad \text{for all } x$$

$$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n\left(1-4^n\right)}{(2n)!} x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots \quad \text{for } |x| < \frac{\pi}{2}$$

$$\sec x = \sum_{n=0}^{\infty} \frac{(-1)^n E_{2n}}{(2n)!} x^{2n} = 1 + \frac{x^2}{2} + \frac{5x^4}{24} + \cdots \quad \text{for } |x| < \frac{\pi}{2}$$

$$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \cdots \quad \text{for } |x| \le 1$$

$$\arccos x = \frac{\pi}{2} - \arcsin x = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = \frac{\pi}{2} - x - \frac{x^3}{6} - \frac{3x^5}{40} - \cdots \quad \text{for } |x| \le 1$$

$$\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \quad \text{for } |x| \le 1,\ x \neq \pm i$$

```
def map(f,seq):
    output = []
    for i in seq:
        output.append(f(i))
    return output
```

Cosine

- Note the sequence

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \quad \text{for all } x$$

n = 0 n = 1 n = 2

- The function is
  - `cf = lambda n:  (x**(2*n) * ((-1)**n)) / factorial(2*n)`

angle

- Just map the function `cf` to

`[0,1,2,3,4,5...]`

- Our target should be
  - `cf(0) + cf(1) + cf(2) + cf(3) + cf(4) + ...`

Cosine

- The function is
  - `cf = lambda n:  (x**(2*n) * ((-1)**n)) / factorial(2*n)`

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \quad \text{for all } x$$

- Our target should be
  - `cf(0) + cf(1) + cf(2) + cf(3) + cf(4) + ....`

```
def myCos(x):
    def cf(n):
            return (x**(2*n) * ((-1)**n) / factorial(2*n))
    return sum(map(cf,range(0,10)))
>>> myCos(3.141592654/3)
0.4999999998815835
>>> cos(3.141592654/3)
0.4999999998815835
```

Sine

- The function is
  - `sf = ???`

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \quad \text{for all } x$$

- Our target should be
  - `sf(0) + sf(1) + sf(2) + sf(3) + sf(4) + ...`

```
def mySin(x):
    def sf(n):
        ????????????
    return sum(map(sf,range(0,10)))
>>> mySin(3.141592654/3)
0.8660254038528065
>>> sin(3.141592654/3)
0.8660254038528065
```

Try other functions (e.g. log, e^x)!

- Can we use our map() for Taylor series?

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \qquad \text{for all } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots \qquad \text{for all } x$$

$$\tan x = \sum_{n=1}^{\infty} \frac{B_{2n}(-4)^n\left(1-4^n\right)}{(2n)!} x^{2n-1} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \cdots \qquad \text{for } |x| < \frac{\pi}{2}$$

$$\sec x = \sum_{n=0}^{\infty} \frac{(-1)^n E_{2n}}{(2n)!} x^{2n} = 1 + \frac{x^2}{2} + \frac{5x^4}{24} + \cdots \qquad \text{for } |x| < \frac{\pi}{2}$$

$$\arcsin x = \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = x + \frac{x^3}{6} + \frac{3x^5}{40} + \cdots \qquad \text{for } |x| \le 1$$

$$\arccos x = \frac{\pi}{2} - \arcsin x$$

$$= \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1} = \frac{\pi}{2} - x - \frac{x^3}{6} - \frac{3x^5}{40} - \cdots \qquad \text{for } |x| \le 1$$

$$\arctan x = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \qquad \text{for } |x| \le 1,\ x \neq \pm i$$

Reduce

reduce()

- What does the following do?

```
>>> reduce(lambda x,y:x+y,[1,2,3])
6
```

```
def reduce(f,seq):
    if not seq:
        return seq
    first = seq[0]
    for i in seq[1:]:
        first = f(first,i)
    return first
```

- first = seq[0] (first == 1)
- for i in [2,3]:
  - i = 2:
    - first = 1 + 2
  - i = 3
    - first = 3 + 3
- return first (first == 6)

reduce()

- In general, let f be a function that takes two arguments

```
reduce(f,[1,2,3,4,5,6])
```

⬇️

```
f(f(f(f(f(1,2),3),4),5),6)
```

- If f is the addition function

```
reduce(f,[1,2,3,4,5,6])
```

⬇️

```
(((((1+2)+3)+4)+5)+6)
```

```
def reduce(f,seq):
    if not seq:
        return seq
    first = seq[0]
    for i in seq[1:]:
        first = f(first,i)
    return first
```

reduce()

```
>>> reduce(lambda x,y:x+y,[1,2,3])
6
```

- This the summation function:

$$\sum_{i=0}^{i<n} seq[i] = seq[0] + seq[1] + seq[2] \ldots$$

```
def reduce(f,seq):
    if not seq:
        return seq
    first = seq[0]
    for i in seq[1:]:
        first = f(first,i)
    return first
```

- How to write the following?

$$\prod_{i=0}^{i<n} seq[i] = seq[0] \times seq[1] \times seq[2] \ldots$$

  - Try it yourself? reduce(lambda x,y:x*y, [1, 2, 3, 4, 5, 6, 7])
  - [1,2,3,4,5,6,7] → 5040

The History of `reduce()`

- In 1994, `reduce()` was a *built-in* function for Python
- Around 2016, `reduce()` was moved to a package called `functools`
  - The fate of reduce() in Python 3000

```
>>> from functools import reduce
>>> reduce(lambda x,y:x+y,[1,2,3])
6
>>> reduce(lambda x,y:x+y,(1,2,3))
6
>>> reduce(lambda x,y:x+y,('a','b','c'))
'abc'
```

- In the same document, you can see that two convenient functions were added
  - any(), all()

any() and all()

- If L = [1,2,3,4], is there ***any*** number that is greater than 3 in L?

```
>>> L = [1,2,3,4]
>>> any(x > 3 for x in L)
True
```

- If L = [1,2,3,4], is there ***any*** number that is greater than 9 in L?

```
>>> any(x > 9 for x in L)
False
```

- Is there ***any*** prime number in the following lists?

```
>>> any(isPrime(x) for x in [4,6,8,9,99])
False
>>> any(isPrime(x) for x in [4,6,8,9,97,99])
True
```

any() and all()

- If L = [1,2,3,4], are ***all*** numbers in L greater than 3 ? (and 0?)

```
>>> all(x > 3 for x in L)
False
>>> all(x > 0 for x in L)
True
```

- Are ***all*** the numbers in the following lists prime numbers?

```
>>> all(isPrime(x) for x in [4,6,8,9,99])
False
>>> all(isPrime(x) for x in [3,5,7,11,97])
True
```