Week 9

Map, Filter, Reduce

Higher order functions

WHY???

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

Expressions	Output
`map(lambda x: x > 2, L)`	[True, False, False, True, True, True, True]
`list(filter(lambda x:x>2,L))`	[9, 3, 4, 5, 6]
`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(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]'

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 A = map(lambda x : x*x , L)

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

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

Convert the function to return a tuple?

- List n =2

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

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

Terminating Condition: input seq is empty

Scale first element

Execute seqScaleR on the rest of the sequence

- Tuple?
  - Try it for 5 min
  - No need to code from scratch, copy the list version and modify
  - Lists cannot be used **at all**

How to convert the functions for tuple?

- List
  - What needed to be changed?

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

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

- Tuple

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

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

How to Sum Digits (Not square yet)?

```
>>> digitsum(22222)
10
```

- Can we use map()?
  - But map only applies on list?
- Hint:
  - Given an integer N, what is

`list(str(N))`

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

```
>>> 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'
```

- Almost?
  - How?

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

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

- How to sum digit squares using map() and sum()?

```
>>> sds(22222)
20
```

  - Try it for 5 mins

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

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

- How to sum digit squares using map() and sum()?

```
>>> sds(22222)
20
```

  - Try it for 5 mins

```
def sds(n):
    return sum(map(lambda x:int(x)**2, list(str(n))))
```

Challenge!

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

Taylor Series with map()

- Can we use it for Taylor series?

Use higher order functions for this?

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

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

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

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

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

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

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

Try cos(x)?

- 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 (x is the angle)
  - `cf = lambda n:  (x**(2*n) * (-1)**n)) / factorial(2*n)`
- 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) + …`

Try cos(x)?

- The function is (x is the angle) L = [0,1,2,3,4,5]
  - `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
```

Try sin(x)?

- The function is (x is the angle)
  - `sf = ???`

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

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

```
def mySin(x):
    def sf(n):
            return (x**(2*n+1) * ((-1)**n) / factorial(2*n+1))
    return sum(map(sf,range(0,10)))
>>> mySin(3.141592654/3)
0.8660254038528065
>>> sin(3.141592654/3)
0.8660254038528065
```

Alternative method

Function for reduce

Function for map

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

Function for filter

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

```
def mapfn(n):
    return -1**n / factorial(2*n+1) * x**(2*n+1)

def filterfn(x):
    return True

def reducefn(x1,x2):
    return x1 + x2

def sin(x, n):
    return reduce( reducefn, map( mapfn, filter(filterfn, range(n+1)) ) )
```

Example of `reduce()`:

python Copy code

```python
from functools import reduce

# Example of summing numbers in a list
numbers = [1, 2, 3, 4, 5]

# Define a function to sum two numbers
def add(x, y):
    return x + y

# Use reduce to sum the numbers
result = reduce(add, numbers)
print(result)  # Output: 15
```

Step-by-step process:

1. `reduce(add, [1, 2, 3, 4, 5])`:
   - First, `add(1, 2)` → returns `3`.
   - Next, `add(3, 3)` → returns `6`.
   - Then, `add(6, 4)` → returns `10`.
   - Finally, `add(10, 5)` → returns `15`.

At the end, the list `[1, 2, 3, 4, 5]` has been reduced to the single value `15`.

And try other TS! And also log, e^x, and so on!

- Can we use it for Taylor series?

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

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

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

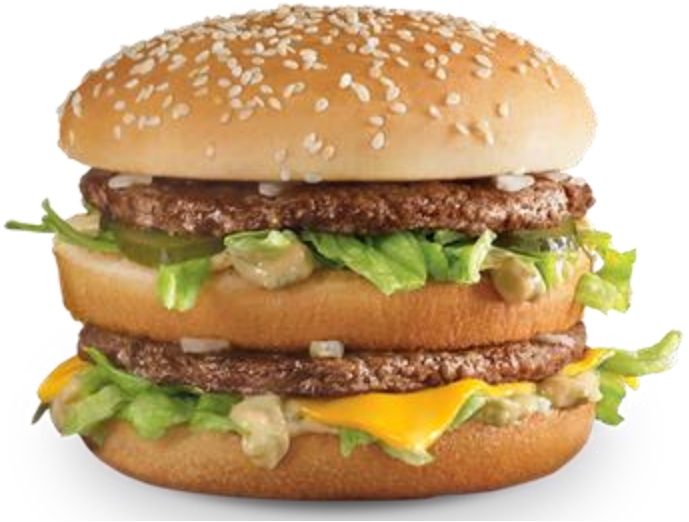
$$\sec x = \sum_{n=0}^{\infty} \frac{(-1)^n E_{2n}}{(2n)!} x^{2n} = 1 + \frac{x^2}{2} + \frac{5x^4}{24} + \cdots \quad \text{for } |x| < \frac{\pi}{2}$$

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

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

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

Remember Burger Price?



From Tutorial 3

- Final version from Tutorial 3

```
def burgerPrice(burger):
    price = 0
    for char in burger:
        if char == 'B':
            price = price + (0.5)
        elif char == 'C':
            price = price + (0.8)
        elif char == 'P':
            price = price + (1.5)
        elif char == 'V':
            price = price + (0.7)
    return price

print(burgerPrice('BVPB'))
```

- Can we use map?
- Intuitively, these are a function *mapping an alphabet to a price* (float)

Burger Price Map Version

```
def price(char):
    if char == 'B':
        return 0.5
    elif char == 'C':
        return 0.8
    elif char == 'P':
        return 1.5
    elif char == 'V':
        return 0.7
    return 0

def burgerPriceM(burger):
    return sum(map(price,burger))
```

Why there is no need to convert the string burger into a list first?

Burger Price Map Version 2

```
def price(char):
    pricedict = {'B':0.5, 'C':0.8, 'P':1.5, 'V':0.7}
    return pricedict[char]

def burgerPriceM(burger):
    return sum(map(price,burger))

print(burgerPriceM('BVPB'))
```

Burger Price Version 3?

- Can we do this directly?

```
def burgerPriceM(burger):
    return sum(map({'B':0.5, 'C':0.8, 'P':1.5, 'V':0.7},burger))
```

- But we can do this:

```
def burgerPriceM(burger):
    return sum(map(lambda x: {'B':0.5, 'C':0.8, 'P':1.5, 'V':0.7}[x],burger))
```

- One line function for `burgerPrice()`!

About one-line function

- Meaning, the function has only one line excluding the function definition and without semi-colon (or other cheating like ‘exec’)
- It’s a cool thing, for example, the following one-liner returns the longest word in a text with commas and periods:

```
>>> def longest(s):
        return sorted(s.replace(',','').replace('.','').split(),key=len)[-1]

>>> print(longest('This is an example text testing the longest() method. The ana
lyzed portion of the text may be of a different shape and punctuation. Regardles
s of that, it should work properly.'))
punctuation
```

- However, do not abuse it
  - Sometime it’s not readable
  - And introducing bugs

Reduce()

(and reuse, recycle?)

Observe the Function "reduce()"

- What does this 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 first = f(first = 1,2)
  - i = 3
    - first = 3 + 3 first = f(first= f(1,2),3)

first = f(first = f(f(1,2),3),4)

- return first (first == 6)

Observe the Function "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)
```

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

- If f is the addition function

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

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

reduce(lambda x,y : x-y, [1,2,3,4])
(((1-2)-3)-4)
reduce(lambda x,y : y-x, [1,2,3,4])
(4-((3-(2-1)))

Obverse the Function “reduce()”

```
>>> 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
```

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'
```

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

- In the same document, you can see that two convenient functions was 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 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 lists are 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
```

More One-liners

- Check Palindrome

```
phrase.find(phrase[::-1])
```

- Print out a file

```
[line.strip() for line in open(filename)]
```

- Factorial:

```
reduce(lambda x, y: x * y, range(1, n+1))
```

- Sieve of Eratosthenes (Finding all prime numbers <= n)

```
set(range(1,n))-set([j for i in range(2,int(n**0.5+1)) for j in range(i*2,n,i)])
```