CSE 30 4-8-21

- email: Please place cse 30 in your subject line.
- Pa1: last ~~ext. to fri.~~ 10 PM
- Pa2: ~~Posted~~ soon
- lab1: due monday

Generators

An iterator can only be used once

```
I = iter(L)
for x in I:
    print(x)
next(I) # StopIteration exception
```

What is a for loop doing 'under the hood':

```
for x in L:
    print(x)
```

is equivalent to:

```
I = iter(L)
x = next(I, None)        # sentinel value
while x != None:
    print(x)
    x = next(I, None)
```

The <u>in</u> operator can be used on an iterator, consuming some or all of the elements.

```
I = iter(L)  # recall L = [1, 2, 3]
1 in I  # True
next(I) # 2
1 in I  # False
next(I) # StopIteration
```

Can also turn an iterator into an iterable

```
I = iter(L)
S = set(I)
print(S) # {1, 2, 3}
next(I) # StopIteration
        # now consumed.
```

You only use what's left in iterator to to construct iterable.

```
I = iter(L)
next(I) # 1
S = set(I) # {2, 3}
```

or

```
I = iter(L)  # recall L = [1, 2, 3]
4 in I # False
S = set(I) # {}
```

A Generator is a type of iterator that can be taylored to specific needs. One way to create a generator is a Generator Expression

Ex. `G = (x**2 for x in range(4))`

```
next(G) # 0
next(G) # 1
next(G) # 4
next(G) # 9, now exhausted
```

when used to construct an iterable (list, set, tuple, ...) we call this <u>comprehension</u>

```
L = [x**2 for x in range(3)]
  # [0, 1, 4]
```

or `L = list( gen. exp. )`

`S = { ... gen. exp. }`

`= set( )`

`T = tuple( )`

we can use logic in generator expressions.

Ex.

```
G = (x**2 for x in range(8) if x%2==1)

# seq. generated:
# 1, 9, 25, 49
```

we can write a function that returns a generator object.

```
def odd_squares(n):
    return (x**2 for x in range(n) if x%2==1)

G1 = odd_squares(8) # different
G2 = odd_squares(8) # gen objects
```

the yield statement:
use in a fcn to create a logically complicated gen. object

```
def gen_fcn():
    .
    .
    .
    yield expression
    .
    .
    .
# end
```

each execution of a yield statement Produces 1 term in the generator object to be returned.

Examples

- LinearSearch.py
- Generator.py