

On your way in...

Pick-up:

1. HW03, graded
2. POGIL Activity 21: Generators

Note:

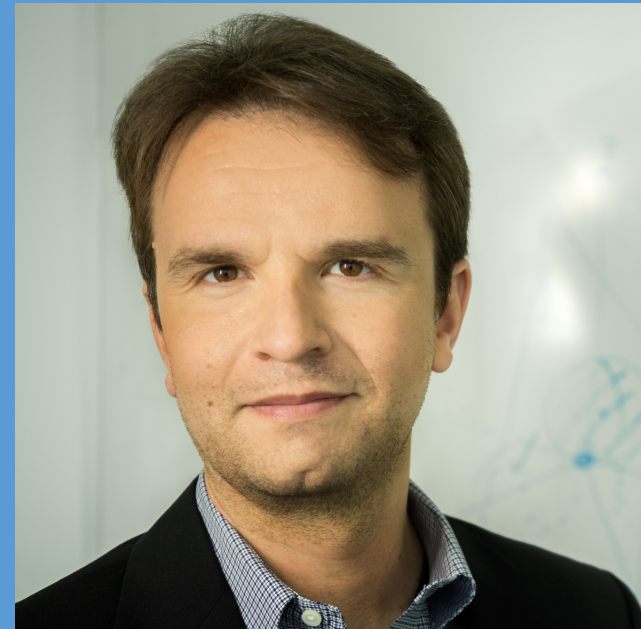
Lab 5 will be released today. It's another partners lab.



CS COLLOQUIUM: NATE DERBINSKY

ADVENTURES IN HYBRID ARCHITECTURES FOR INTELLIGENT SYSTEMS

Today at 2:35pm in Wege (TCL123)
There's snacks!



Midterm Exam is Thursday, March 12

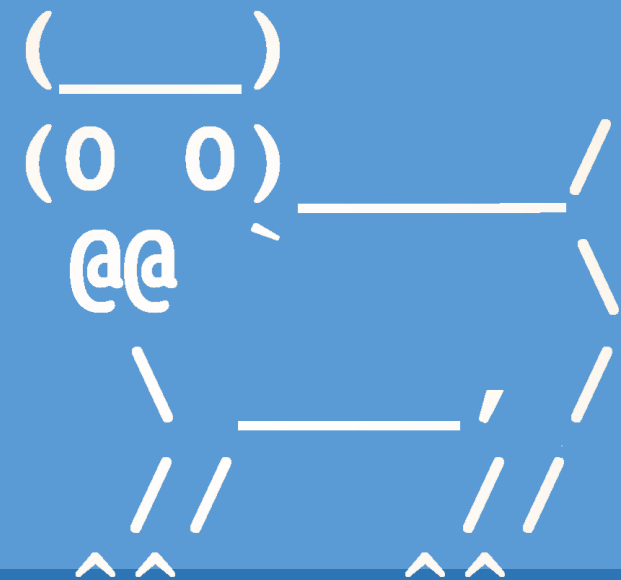
- TPL 203: 5:45pm-7:45pm OR 8-10pm
- **Exam Review Session: 3/9 at 7-8:30pm in TPL 203.**
- Closed book exam
- Review your homeworks! POGILs! Slides! Labs!
- Next week's lab will be less intense



Welcome to CS 134!

Introduction to Computer Science
Iris Howley

-Generators & Plotting-



HW4 #2 Typo

- `collegeInfo = ["Williams College", 1793, "MA"]`
- `colName = collegeInfo[0]`
- `foundYear = collegeInfo[1]`
- `colState = collegeInfo[2]`



TODAY'S LESSON

Plotting Data

(Gathering data & producing graphs with it!)

matplotlib

- A plotting module: `from matplotlib import pyplot as plt`

- Needs a list of x-values and a list of y-values

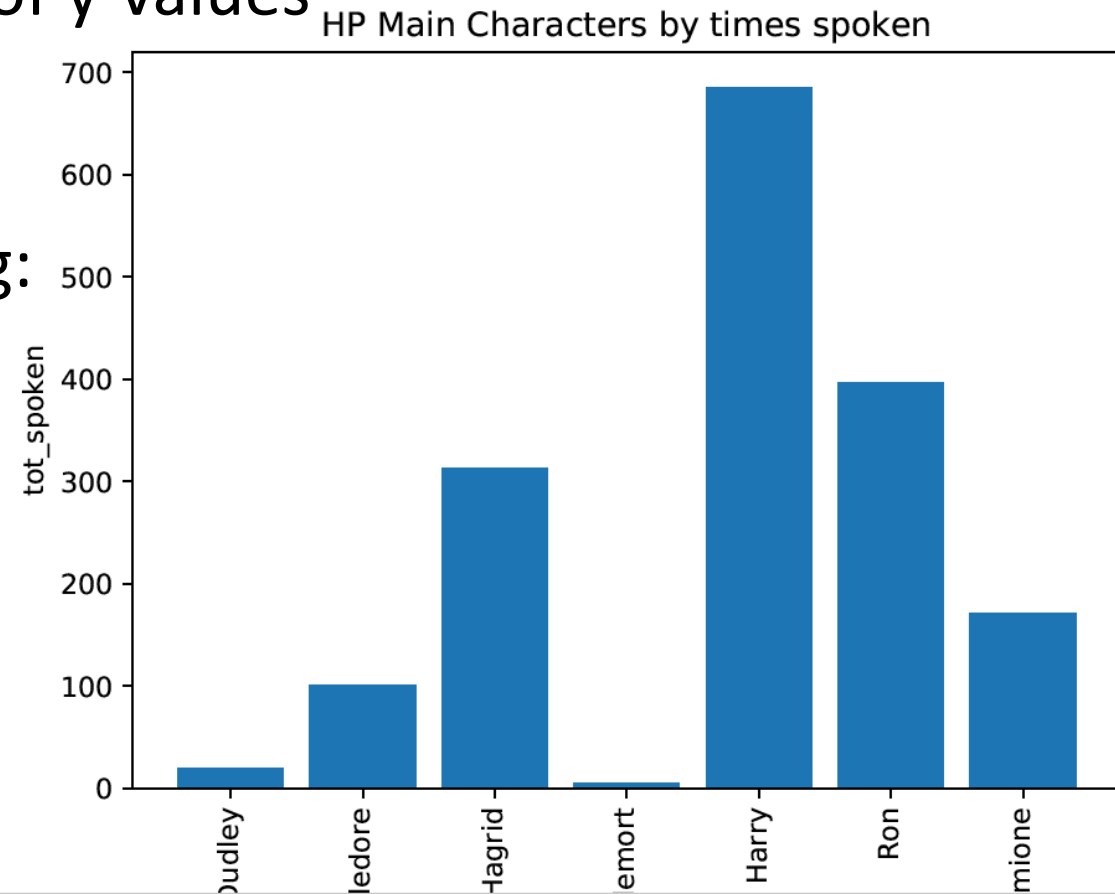
- `xLabels = []`

- `yValues = []`

- Number of x-values is useful for spacing:

- `positions = list(range(len(xLabels)))`

- Then, we plot!



matplotlib

`# plt.figure` makes our figure

- `plt.figure()` `# plt.bar` determines what kind of plot
- `plt.bar(positions, yValues)`
- `plt.xticks(positions, xLabels, rotation=90)` `# xticks` helps w/ X label positions
- `plt.title('plot title goes here')`
- `plt.xlabel('x-axis label goes here')`
- `plt.ylabel('put your y-axis label here')`
- `plt.tight_layout()` `# Ensures all our labels, etc fit`
- `plt.savefig('filename.pdf')` `# To save plot to a file!`
- `plt.show()`

More matplotlib features in their documentation: http://matplotlib.org/users/pyplot_tutorial.html

An Example

- Read in the Harry Potter Data
- Make a list of dictionaries [
 - {NAME: <character name>, TOTAL:<tot times spoken>, WORDS: <list words said>}
 - {NAME: 'Harry', TOTAL:685, WORDS: [...,'magic',...'wizard',...]}
- Go through list of dictionaries, make a list of just character names and a list of just total times spoken
- Make a bar chart with X being character and Y being number of times spoken
- ...How would we sort this?

See Example Code on Website



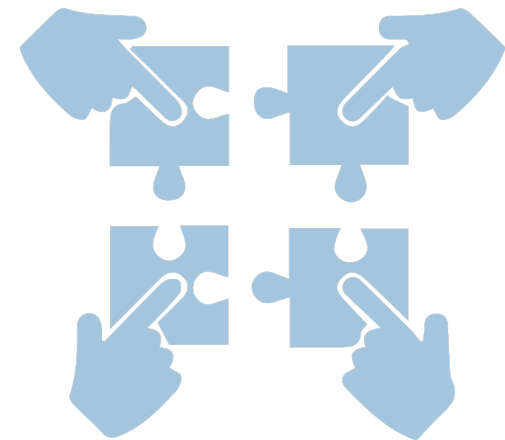
TODAY'S LESSON

Generators

(A memory efficient way of generating on-demand values)

POGIL Activity 21 - Generators

- Look at Python Activity 21, Question 1-4 & 6 (5 on your own)
- Find a partner and talk through the questions together

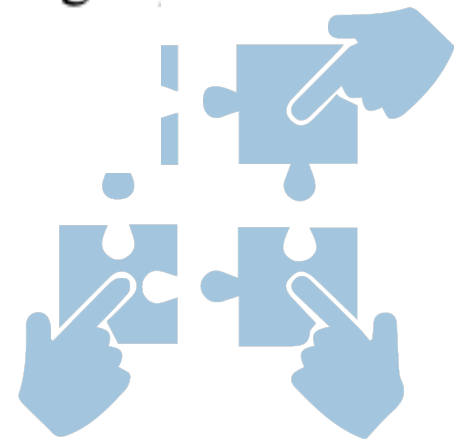


POGIL – Activity 21: Question 1

```
0 >>> def countEvens(n):
1 ...     i = 0
2 ...     while i <= n:
3 ...         print(i)
4 ...         i += 2
5 >>> countEvens(3)
```

What another way we can write this while loop?

- When does the `while` loop on line 2 stop? _____
- If the parameter `n` was 3, how many times through the loop would we go?
- What is the output from calling `countEvens`, on line 5?



POGIL – Activity 21: Question 2

```
0 >>> def countEvens(n):
1     ...     i = 0
2     ...     while i <= n:
3     ...         yield i
4     ...         i += 2
5 >>> g = countEvens(3)
6 >>> print(next(g))
7     0
8 >>> print(next(g))
9     2
10 >>> print(next(g))
```

- How does the function `countEvens(n)` differ from the previous `countEvens(n)`?

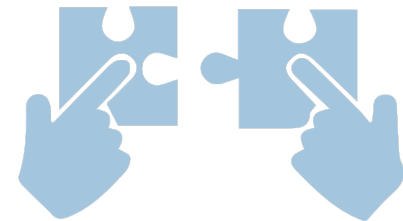
- If we replace line 5 with `g = countEvens(6)`, what will the output of line 10 be?

- Write a line of code to print the next value yielded by `g`.

- With line 5 as `g = countEvens(3)`, what might the output of line 10 be?

- With `g = countEvens(3)`, line 10 will produce a `StopIteration` exception. Why might that be?

- Write a new function, `reverseGen(...)`, that takes a list and yields values from the list from the end to the beginning:
`def reverseGen(mylist):`



Generators Syntax

```
def countEvens (n) :
```

```
    i = 0
```

```
    while i < n: yield i
```

```
        yield i
```

```
        i+= 2
```

keyword, is like return, but lets you return to the function and continue where you left off!
This is where we'll pick up again 2nd time we call next on the

generator

Variable to store our generator object

```
g = countEvens (3)
```

Function that yields, is a generator object.

```
print (next (g) )
```

Pass 3 as an argument, will be called n in function

```
next (g)
```

How we ask the generator to yield the next item in the sequence

Returns to countEvens and starts at the i+=2 line

POGIL – Activity 21: Question 3

```
0 >>> def countEvens(n):
1 ...     i = 0
2 ...     while i <= n:
3 ...         yield i
4 ...         i += 2
5 >>> for num in countEvens(3):
6 ...     print(num)
```

a.

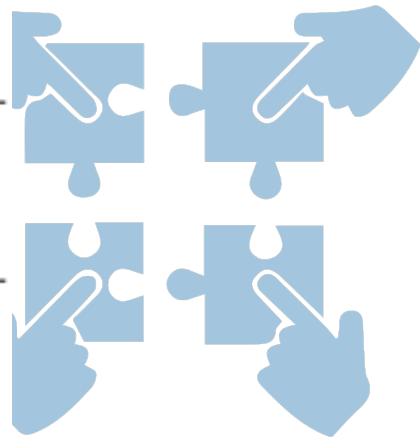
The output from this sample code is the same as the output from Question 1. What might the `for..` loop be doing in order to make this possible?

b.

What will this code output?

c.

Write a couple lines of code to use your `reverseGen(..)` generator from the previous question, using a `for..` loop:



POGIL – Activity 21: Question 4

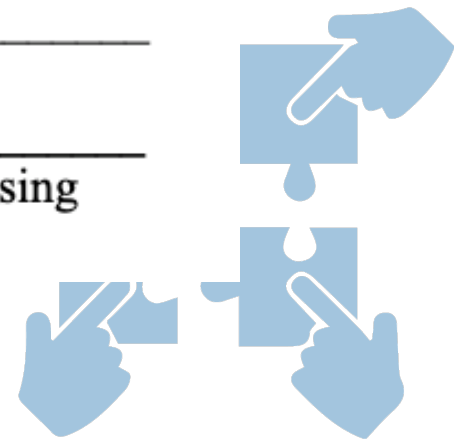
4. Examine the following Python code:

```
0 def count(start = 0, step = 1):  
1     i = start  
2     while True:  
3         yield i  
4         i += step
```

- a. How do the parameters of this `count(..)` function differ from those of `countEvens(..)`?

- b. If we wanted to replicate the behavior of `countEvens(..)` with the `count(..)` function, what would our `start` and `step` values be?
`start:` _____ `step:` _____
- c. When does the `while` loop on line 2 end?

- d. Write a few lines of code to output the first four multiples of the number three using `count(..)`:



POGIL – Activity 21: Question 5

5. Examine the following code from interactive python:

```
0 >>> def letters(word, n):
1 ...     i = 0
2 ...     while i < n:
3 ...         yield word[i]
4 ...         i += 1
```

```
5 >>> g=letters('good', 3)
6 >>> next(g)
7 'g'
8 >>> next(g)
9 'o'
10 >>> next(g)
11 'o'
12 >>> next(g)
13 Traceback ():
    Line 1 in <module>
Stop Iteration
```

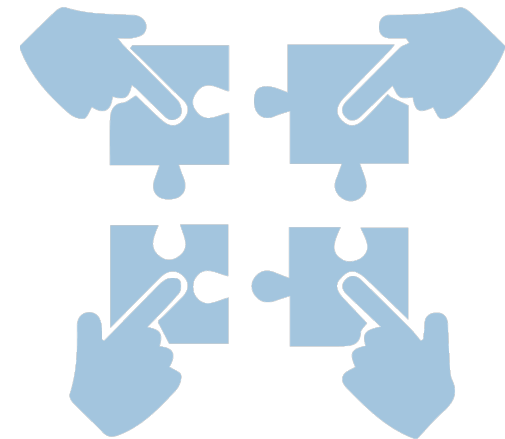
a. What does the `letters(word,n)` function do?

- b. What are the values of the arguments passed to `letters(..)` on line 5? ___
- c. What does the calls to `next(g)` do on lines 7, 9, and 11?
-

d. Why might an error have been thrown by the `next(g)` call on line 12?

e. What would happen if we replaced line 5 with `g=letters('good', 4)`?

f. What might happen if we replaced line 5 with `g=letters('bye', 4)`?



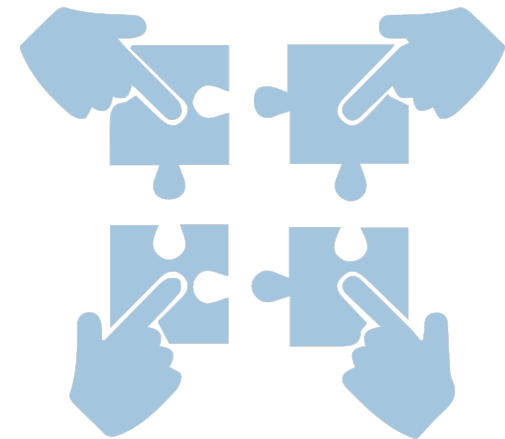
POGIL – Activity 21: Question 6

```
def mystery(a = 0, b = 1):  
    yield a  
    yield b  
    while True:  
        a, b = b, a+b  
        yield b
```

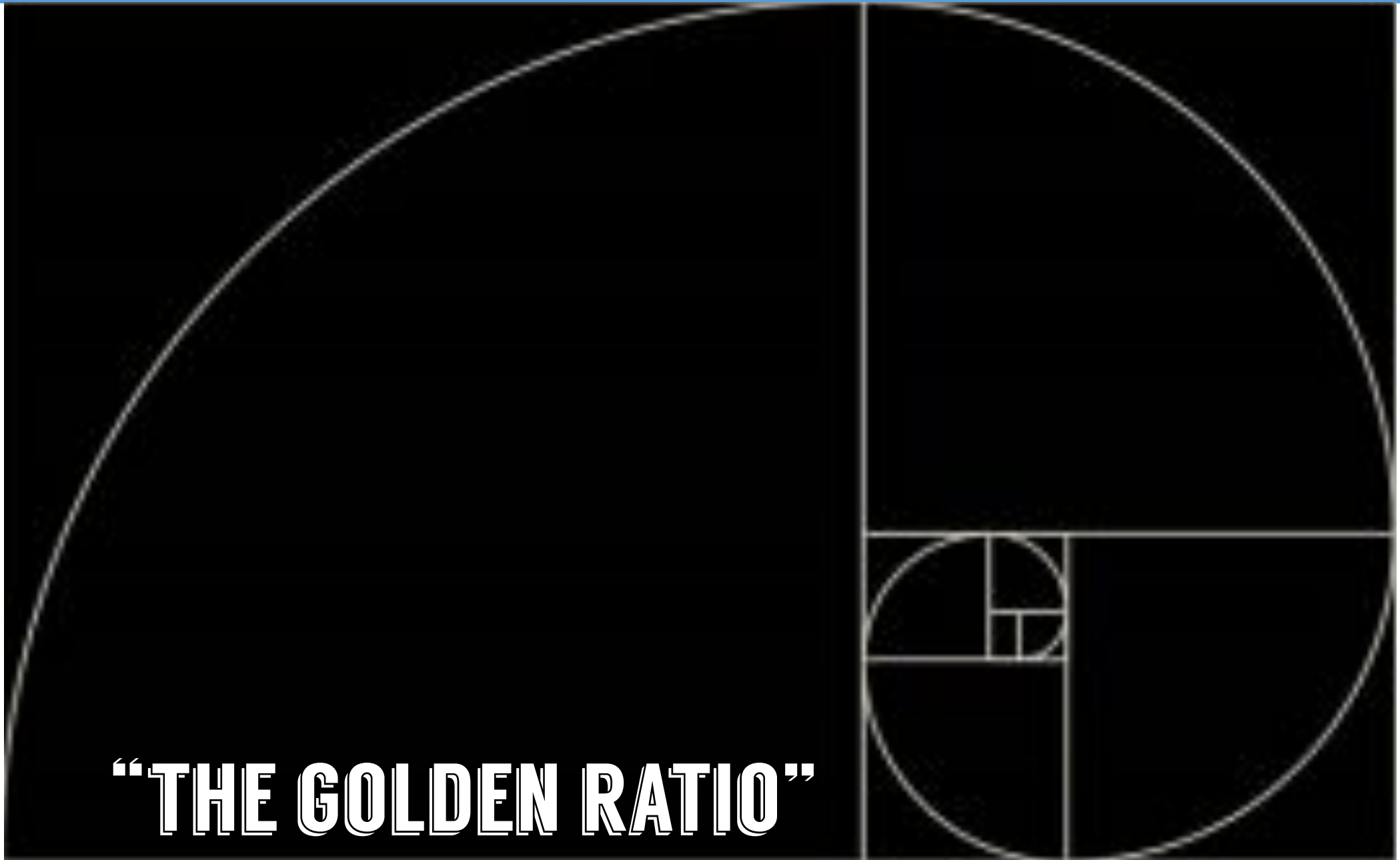
- e. Use the following table to step-through what this function is doing:

a	b	Yield Statement	Yielded
0	1	yield a	0
0	1	yield b	1
1	1	yield b (2)	1
1	2	yield b (2)	

- b. If we were to rename this function to something more meaningful, what would we name it to?



“THE GOLDEN RATIO”





WIKIPEDIA
The Free Encyclopedia

main page
contents
featured content
current events
random article
donate to Wikipedia
Wikipedia store
interaction

help
about Wikipedia
community portal
recent changes
contact page

Article

Talk

Read

Edit

View history

Search Wikipedia



Fibonacci number

From Wikipedia, the free encyclopedia

"Fibonacci Sequence" redirects here. For the chamber ensemble, see [Fibonacci Sequence \(ensemble\)](#).

In [mathematics](#), the **Fibonacci numbers**, commonly denoted F_n , form a [sequence](#), called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from 0 and 1.

That is,^[1]

$$F_0 = 0, \quad F_1 = 1,$$

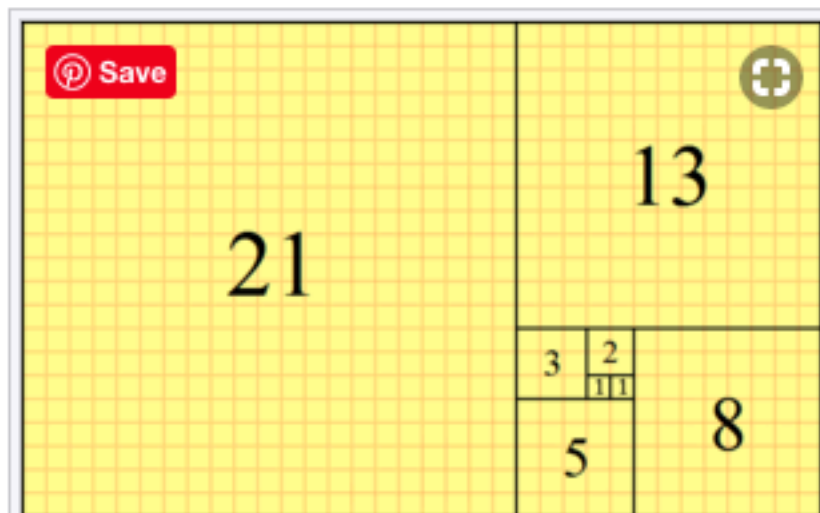
and

$$F_n = F_{n-1} + F_{n-2},$$

for $n > 1$.

The beginning of the sequence is thus:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...^[2]

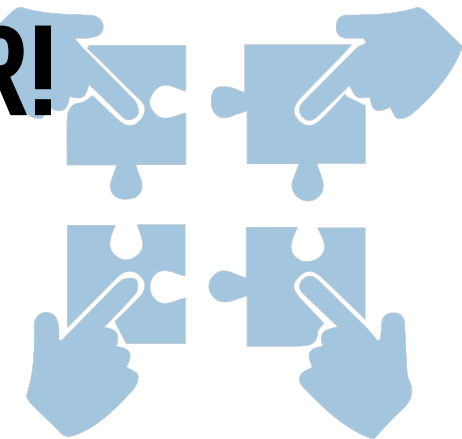


A tiling with squares whose side lengths are successive Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13 and 21.

**YOU SHOULD COMPLETE THE REST OF
ALL POGILS OUTSIDE OF CLASS.**

BEST DONE WITH A PARTNER OR STUDY GROUP.

CHECK YOUR ANSWERS ON A COMPUTER!



TODAY'S LESSON

Generators

(A memory efficient way of generating on-demand values)

Generators

```
def countTo(n):  
    i = 1  
    while i <= n:  
        yield i  
        i += 1
```

```
g = countTo(3)          print(next(g))          3  
print(next(g))         1  print(next(g))  
print(next(g))         2  ERROR StopIteration
```


Generators


```
def countTo(n):  
    i = 1  
    while i <= n:  
        yield i  
        i += 1
```

```
g = countTo(3)  
print(next(g))    1  
print(next(g))    2
```


```
def countRet(n):  
    i = 1  
    while i <= n:  
        return i  
        i += 1
```

```
print(countRet(5))    1  
print(countRet(5))    1  
print(countRet(5))    1
```

Can have multiple return statements

```
def countRet(n):  
    i = 1  
    while i <= n:  
         return i  
        i += 1
```

Once we reach 'return'
we never get past it!
i is never incremented!

```
def multRet(num):  
    if num <= 0:  
        return num  
    else:  
         return "+++"
```

"+++" is only returned if
"return num" is never
reached, i.e., when num
is greater than 0.

Another example

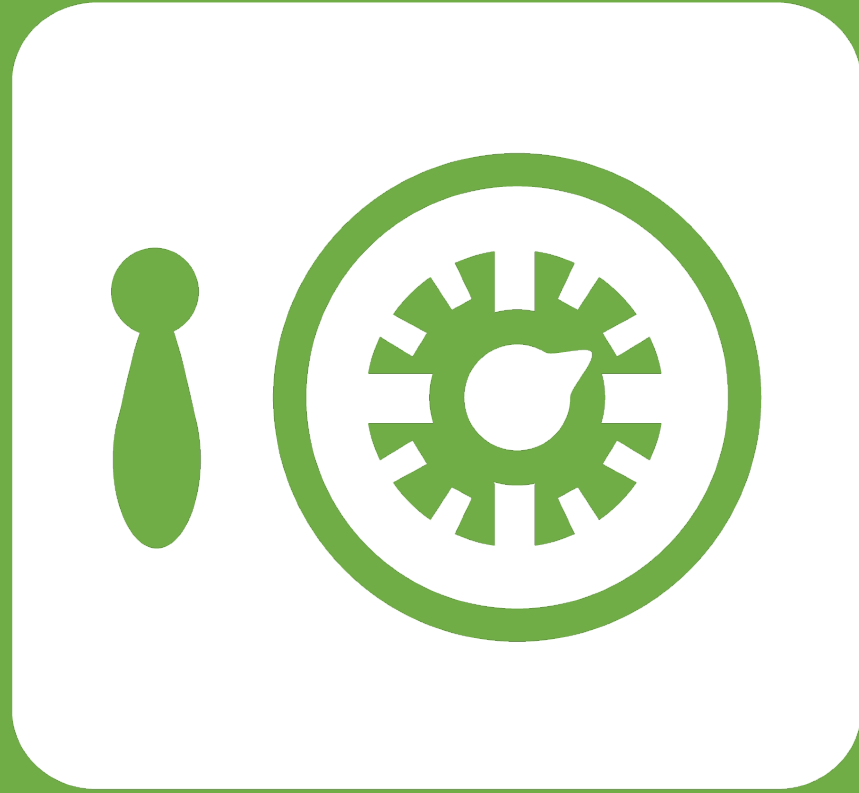
```
def primes():
    """Generates all primes."""
    p = 2
    while True:
        if isPrime(p):
            yield(p)
        p += 1

for i in primes():
    print(i)
```

```
def isPrime(n):
    """Returns True iff n is prime."""
    if n <= 2:
        return n == 2
    g = primes()
    f = next(g)
    while f*f <= n:
        if (n%f) == 0:
            return False
        f = next(g)
    return True
```

QUESTIONS?





Leftover Slides