Name:_____ **Partner:** _____

Python Activity 21: Generators

Learning Objectives
Students will be able to:
Content:
- Define a **generator**
- Explain the difference between the **yield** and return keywords

Process:
- Write code that creates a generator.
- Write code that uses a generator via **next(..)** and a for..loop.

Prior Knowledge
- Python concepts from Activities 1-20.

Folks, this is a brand new activity. If you encounter any issues/typos, please let Iris know!

Critical Thinking Questions:

1. Examine the sample code from interactive python, below.

Sample Code

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

a. When does the `while loop` on line 2 stop? _____

b. If the parameter `n` was 3, how many times through the loop would we go? _____

c. What is the output from calling `countEvens`, on line 5?

_____

FYI: A ***generator*** is an object that constructs a (possibly infinite) stream of values on demand.

2. Examine the sample code from interactive python, below.

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

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

_____

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

_____

c. If we replace line 5 with `g = countEvens(10)`, what will the first five calls of `next(g)` generate? _____ _____ _____ _____ _____

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

```
def reverseGen(mylist):
```

_____

_____

_____

_____

_____

_____

FYI: ***Yield*** is a keyword like return, but instead of returning a value, it surrenders a generator object. The special method ***next(..)*** is required to retrieve the value that was yielded.

3. The following code occurs in interactive Python:

```
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:

_____

_____

_____

_____

FYI: A more efficient mechanism for using generators is by using a for loop.

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(..)`:

_____

_____

_____

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)`?

_____

6. Examine the following Python code:

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

a. 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?

_____

Application Questions: Use the Python Interpreter to check your work

1. Write a function that uses the yield statement to infinitely generate all the odd numbers:

```
def oddNum():
```

_____

_____