

<u>Curso</u> > <u>Week 5</u>... > <u>10. An</u>... > Exercis...

Audit Access Expires 5 de ago de 2020

You lose all access to this course, including your progress, on 5 de ago de 2020.

Exercise 2

Finger Exercises due Aug 5, 2020 20:30 -03

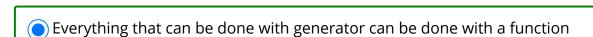
Exercise 2

10/10 points (graded)

ESTIMATED TIME TO COMPLETE: 18 minutes

This problem will ask a series of questions about generators.

1.	Thinking about the <code>genPrimes</code> generator from the last problem, which of the following can be done only by using a generator, instead of defining a function (that uses any type of construct we've learned about, except generators)?		
	Return 1000000 prime numbers		
	Print every 10th prime number, until you've printed 20 of them		
	() Keep printing the prime number until the user stops the program		





Explanation:

We could write a function that does any of the choices. However a generator is nice because we can ask the generator for the next item, one at a time, and we don't waste time computing values that we don't ultimately want (or won't want for a long time).

```
def genPrimesFn():
    '''Function to return 1000000 prime numbers'''
    primes = []
                  # primes generated so far
    last = 1
                  # last number tried
    while len(primes) < 1000000:
        last += 1
        for p in primes:
            if last % p == 0:
                break
        else:
            primes.append(last)
    return primes
def genPrimesFn():
    '''Function to print every 10th prime
    number, until you've printed 20 of them.'''
    primes = [] # primes generated so far
                 # last number tried
    last = 1
    counter = 1
    while True:
        last += 1
        for p in primes:
            if last % p == 0:
                break
        else:
            primes.append(last)
            counter += 1
            if counter % 10 == 0:
                # Print every 10th prime
                print(last)
            if counter \% (20*10) == 0:
                # Quit when we've printed the 10th prime 20 times (ie we've
                # printed the 200th prime)
                return
def genPrimesFn():
    '''Function to keep printing the prime number until the user stops the
program.
    This way uses user input; you can also just run an infinite loop (while
True)
    that the user can quit out of by hitting control-c'''
                  # primes generated so far
    primes = []
    last = 1
                  # last number tried
                  # Assume we want to at least print the first prime...
    uinp = 'y'
    while uinp != 'n':
        last += 1
        for p in primes:
            if last % p == 0:
                break
```

```
else:
           primes.append(last)
           print(last)
           uinp = input("Print the next prime? [y/n] ")
           while uinp != 'y' and uinp != 'n':
                print("Sorry, I did not understand your input. Please enter
'y' for yes, or 'n' for no.")
               uinp = input("Print the next prime? [y/n] ")
```

2. Every procedure that has a yield statement is a generator.



Explanation:

See https://docs.python.org/3/reference/expressions.html?highlight=yield. The Python documentation is always your friend!

3. If a procedure has only one yield statement, but that statement will never be executed, then the procedure is not a generator.

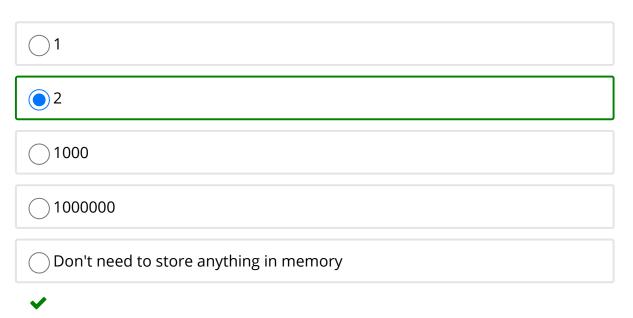
└	
False	
True	

Explanation:

Examine the following code; play around with it in Python.

```
def generator1():
    if True:
        yield 1
def generator2():
    if False:
        yield 1
g1 = generator1()
g2 = generator2()
print(type(g1))
print(type(g2))
print(next(gen))
print(next(gen))
```

4. If we were to use a generator to iterate over a million numbers, how many numbers do we need to store in memory at once?



Explanation:

We need to store 2 numbers - one for the current value, and one for the max value.

```
def genOneMillion():
   maxNum = 1000000
    current = -1
   while current < maxNum:
        current += 1
        yield current
```

Python actually provides this! The range function is a generator.

For the following tasks, would it be best to use a generator, a standard function, or eith

1. Finding the nth Fibonacci number				
Generator				
Standard function				
Either a generator or standard function is fine				
2. Printing out an unbounded sequence of Fibonacci numbers				
Generator				
Standard function				
Either a generator or standard function is fine				
3. Printing out a bounded sequence of prime numbers, where the prime numbers are successively computed by division by smaller primes				
Generator				
Standard function				
Either a generator or standard function is fine				
✓				
4. Printing out an unbounded sequence of prime numbers, where the prime numbers are successively computed by division by smaller primes				
Generator				
Standard function				
<i>*</i>				

)20	Exercise 2 10. An extended example Material didat	11CO 6.UU.1X edX		
	Either a generator or standard function is fine			
5. F	Finding the score of a word from the 6.00x Word Game o	of Pset 4		
	Generator			
	Standard function			
	Either a generator or standard function is fine			
	✓			
	terating over a sequence of numbers in a random order repeated	, where no number is		
	Generator			
Standard function				
	Either a generator or standard function is fine			
	✓			
Envia	ar			
1 An	nswers are displayed within the problem			
Exerci	ise 2	Ocultar discussão		
Горіс: Lec	cture 10 / Exercise 2			
		Add a Post		
Show al	Il posts ✓	por atividade recente 🗸		
<u> </u>	hen to use generator vs standard function? makes sense to me that for unbounded sequences, we should use generate	ors to prevent memory		
	· — ——————————————————————————————————			

? Question 4 part 1 For question 4. If we were to use a generator to iterate over a million numbers, how many numbers ... ? Unbounded vs. Bounded? 2 Can anyone clarify the difference when the question asks about "unbounded" vs. "bounded"? What i...

© All Rights Reserved