



EXPLORER

OPEN EDITORS
- percipio59_Recursive Generators...

PYTHON
- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Looping
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Generators
    - percipio50_Basic Iteration.py
    - percipio51_The map() Function.py
    - percipio52_The Filter() Function....
    - percipio53_The functools.reduce...
    - percipio54_Implementing an Iter...
    - percipio55_Implement an Iterabl...
    - percipio56_Implement an Iterabl...
    - percipio57_Simple Generators.py
    - percipio58_Lazy Generators.py
    - percipio59_Recursive Generators...
    - percipio60_Exercise-Creating an ...
  - 09_Exceptions
  - 10_Automation Programming
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_master
- excel_code_summary_master.py
- PIP_Help-2.PNG
- PIP_Help.PNG
- Python_Clear-Window-Command.txt
- python_debug_logging_code.py
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts.docx
- Python_Tutorials.md

percipio59_Recursive Generators.py

```
'''
percipio59_Recursive Generators.py
Percipio video: Iterables-and-Generators; Recursive Generators

* Demonstrate Recursive Generators
* Recursion - the repeated application of a recursive procedure or definition,
    characterized by recurrence or repetition.
* Generators can be used without filling up RAM allowing use without a limit and able to
    continue on as large as the computer is able to handle the integer size
* Recursion is natural when using generators in Python because generators can be
    repetively-called functions able to hold their previous value.
* Useful when need to create a calculations multiple times, rather then filling up RAM with
    a list, use a generator-object
'''
nl = '\n'
print('Classic Recursive function generating a fibonacci sequence up to "n" elements', nl)
def fib_func(n): # fib_func, a normal function to demonstrate classic recursion
    if n < 2: # if 'n' is less than 2,...
        return n # then simply returns 'n'
    return fib_func(n-2) + fib_func(n-1) # otherwise returns itself with n-2 plus itself
        with n-1
    ''' This shows recursive behavior, it calls itself with a value. This function gets
        called repeatedly and can be used to generate a list of fibonacci numbers. '''

func_list = [fib_func(n) for n in range(10)] # func_list calls fib_func with 'n' for each
    'n' in the range up to, but not including, 10.
print('The func_list of values:', func_list) # prints the func_list of values
func_gen = (fib_func(n) for n in range(10)) # identical to the list comprehension
    expression above, but by using a generator expression () instead, creates a generator
    object.
print('The func_gen of values:', func_gen, nl) # shows the generator object was created
for n in range(10):
    print('The func_gen value:', next(func_gen)) # iterates through func_gen for the range
        of numbers on demand, generating the values only when called

print(nl, 'Recursive generator function (fib_gen(n))', nl)
def fib_gen(n): # generate values with a true generator function by using 'fib_gen' which
    is recursive and a generator.  Function is initially called with an 'n' value which
    determines how many times the while loop is able to execute.
    ''' Recursive generator function: Generate a fibonacci sequence up to n elements
```

EXPLORER

OPEN EDITORS
- percipio59_Recursive Generators...

PYTHON
- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Looping
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Generators
    - percipio50_Basic Iteration.py
    - percipio51_The map() Function.py
    - percipio52_The Filter() Function....
    - percipio53_The functools.reduce...
    - percipio54_Implementing an Iter...
    - percipio55_Implement an Iterabl...
    - percipio56_Implement an Iterabl...
    - percipio57_Simple Generators.py
    - percipio58_Lazy Generators.py
    - percipio59_Recursive Generators...
    - percipio60_Exercise-Creating an ...
  - 09_Exceptions
  - 10_Automation Programming
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_master
- excel_code_summary_master.py
- PIP_Help-2.PNG
- PIP_Help.PNG
- Python_Clear-Window-Command.txt
- python_debug_logging_code.py
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts.docx
- Python_Tutorials.md

percipio59_Recursive Generators.py

```
28      ''' Recursive generator function: Generate a fibonacci sequence up to n elements
29      '''
30      element = 0 # an initially-set-to-zero element counter which keeps track how many to
            generate
31      f1, f2 = 0, 1 # Initialized values set of f1=0 and f2=1
32      while element < n: # only iterate while the element is less than 'n' and
33          yield f1 # while the element is less than 'n', it yields f1
34          f1, f2 = f2, f1 + f2 # then f1 is assigned what f2 was, and f2 is assigned f1+f2
35          element += 1 # each time iterate through, increment by +1
36      ''' This is recursive because in a generator, it's going to continue to execute itself
            repeatedly with the new values.
37      Each time it yields f1, it calculates a new f1 and f2 and is incrementing that element
            by 1.
38      While the element is less then 'n', it yields the new calculated f1 value
39      '''
40
41  gen_fib = fib_gen(10) # assign a variable to the function 'fib_gen' called with a parameter
        of 10
42  print('The gen_fib values:', gen_fib) # print the values of that function-object seen as a
        generator-object directly
43  for n in range(10): # iterate over that function-object using a forLoop with the same
        number used to create the generator to iterate over it showing each generation of a
        value
44      print('The gen_fib value:', next(gen_fib)) #
45
46  print(nl, 'Recursive generator function (fib_inf(n)) without a limit', nl)
47  def fib_inf(): #
48      ''' Recursive generator function: Generate a fibonacci sequence up to n elements
49      '''
50      f1, f2 = 0, 1 # Initialized values set of f1=0 and f2=1
51      while True: # Instead of a limited while loop, this function continues while True,...
52          yield f1 # ... yielding f1 and,...
53          f1, f2 = f2, f1 + f2 # ... calculating f1=f2 and f2=f1+f2
54
55  inf_fib = fib_inf() # create an infinitely capable generator as a new generator-object
56  print('The inf_fib of values:', inf_fib) #
57  for n in range(10): # ability to generate as many times as needed, in this example only 10
58      print('The inf_fib value:', next(inf_fib)) #
59  '''
60  RESULT:
```





EXPLORER

OPEN EDITORS 1 UNSAVED

- percipio59_Recursive Generators...

PYTHON

- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Looping
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Generators
    - percipio50_Basic Iteration.py
    - percipio51_The map() Function.py
    - percipio52_The Filter() Function....
    - percipio53_The functools.reduce...
    - percipio54_Implementing an Iter...
    - percipio55_Implement an Iterabl...
    - percipio56_Implement an Iterabl...
    - percipio57_Simple Generators.py
    - percipio58_Lazy Generators.py
    - percipio59_Recursive Generators...
    - percipio60_Exercise-Creating an ...
  - 09_Exceptions
  - 10_Automation Programming
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_master
- excel_code_summary_master.py
- PIP_Help-2.PNG
- PIP_Help.PNG
- Python_Clear-Window-Command.txt
- python_debug_logging_code.py
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts.docx
- Python_Tutorials.md
- Scripts - Shortcut.lnk

percipio59_Recursive Generators.py ●

```
RESULT:
Classic Recursive function generating a fibonacci sequence up to "n" elements
The func_list of values: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
The func_gen of values: <generator object <genexpr> at 0x0000013135A24C50>
The func_gen value: 0
The func_gen value: 1
The func_gen value: 1
The func_gen value: 2
The func_gen value: 3
The func_gen value: 5
The func_gen value: 8
The func_gen value: 13
The func_gen value: 21
The func_gen value: 34

 Recursive generator function (fib_gen(n))
The gen_fib values: <generator object fib_gen at 0x0000013135A24D00>
The gen_fib value: 0
The gen_fib value: 1
The gen_fib value: 1
The gen_fib value: 2
The gen_fib value: 3
The gen_fib value: 5
The gen_fib value: 8
The gen_fib value: 13
The gen_fib value: 21
The gen_fib value: 34

 Recursive generator function (fib_inf(n)) without a limit
The inf_fib of values: <generator object fib_inf at 0x0000013135A24D58>
The inf_fib value: 0
The inf_fib value: 1
The inf_fib value: 1
The inf_fib value: 2
The inf_fib value: 3
The inf_fib value: 5
The inf_fib value: 8
The inf_fib value: 13
The inf_fib value: 21
The inf_fib value: 34
'''
```

0 ⚠ 0 — Ln 60, Col 1 (7 selected) — Spaces: 4 — UTF-8 — CRLF — Python