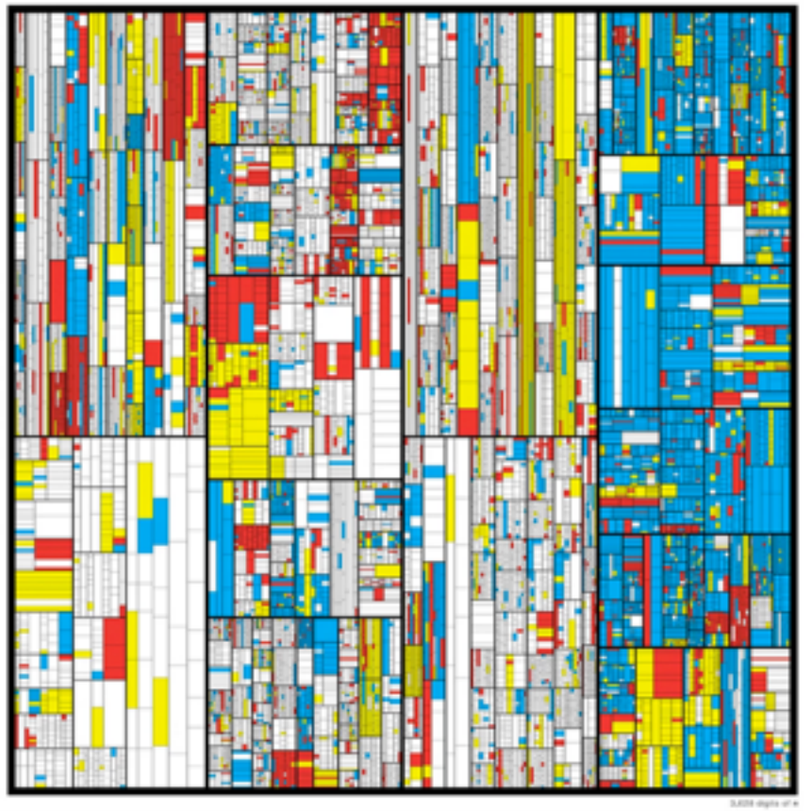
LECTURE 17 & 18

Applications: Visualizing π the Python Way: Iterators, Generators and Queues

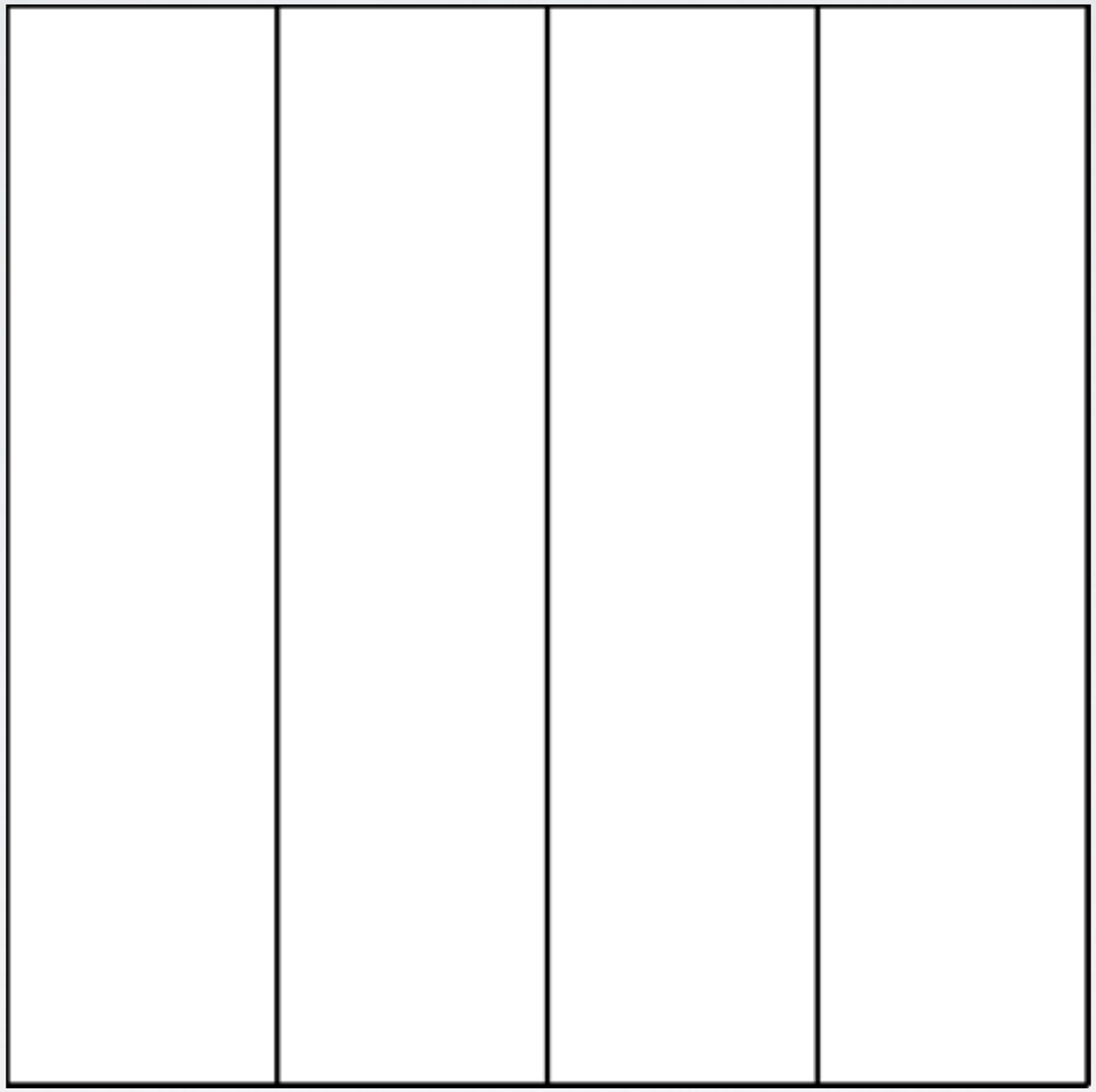
Martin Krzywinski

http://www.washingtonpost.com/blogs/wonkblog/wp/2015/03/14/10-stunning-images-show-the-beauty-hidden-in-pi/

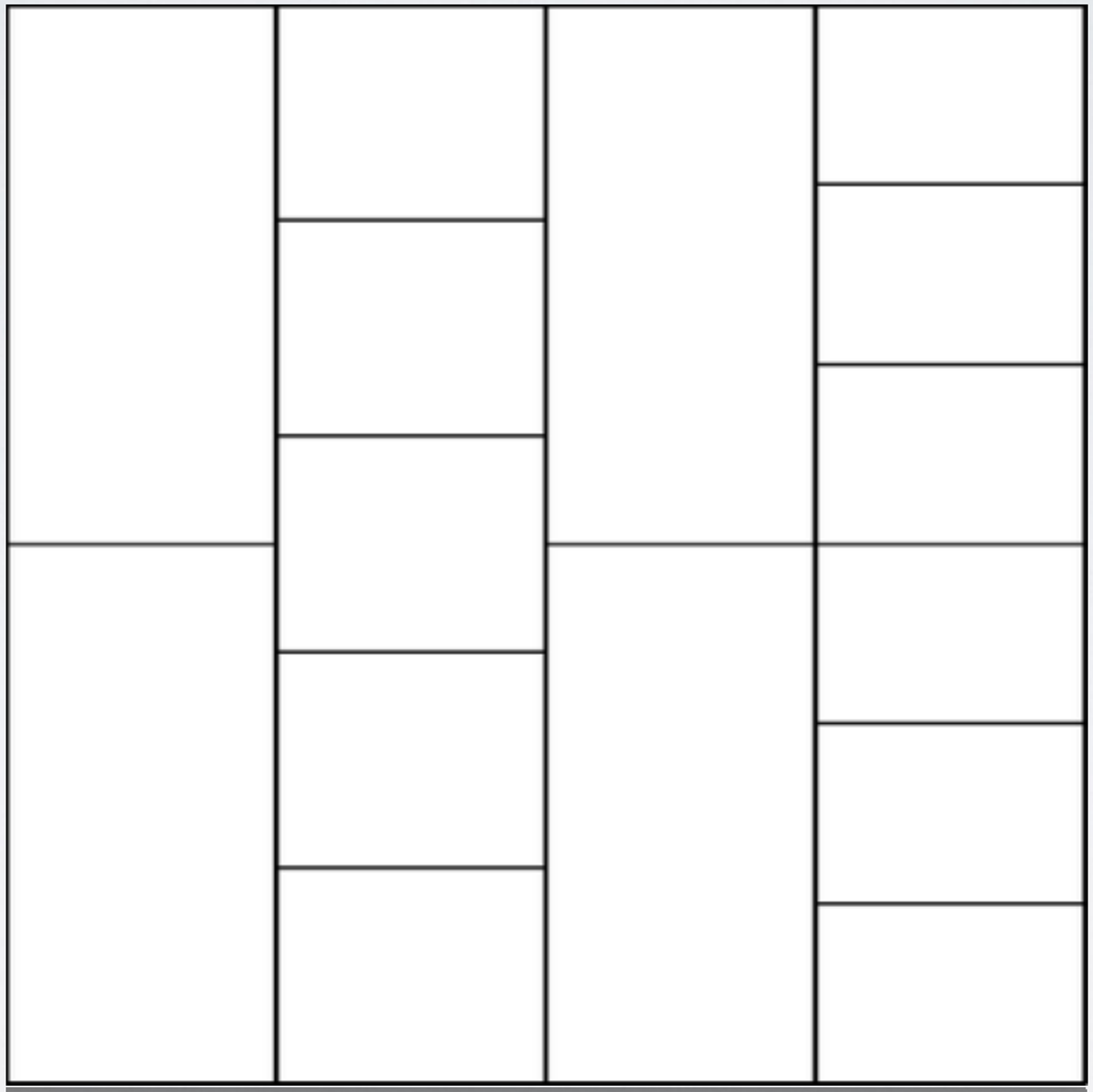
Below is Krzywinski's new illustration for 2015, a type of diagram that is called a treemap. He first divides the box by drawing "3" lines vertically. Then he divides the first box horizontally by drawing "1" line, the second by drawing "4" lines, and so on. Here, Krzywinski randomly colored the graphic with the primary colors used by members of the De Stijl and Bauhaus art movements in the 1920s, like Piet Mondrian, Paul Klee and Joseph Albers.



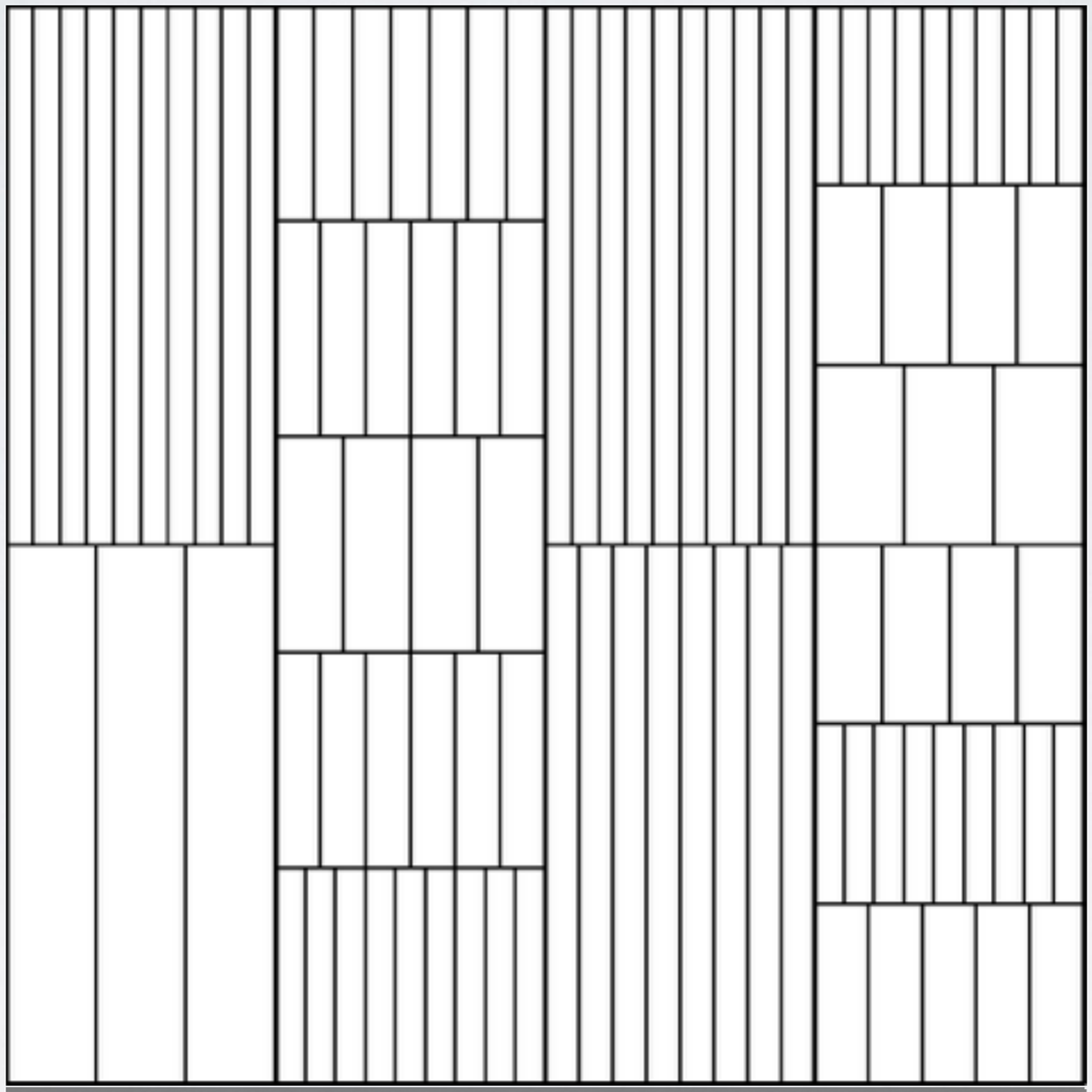
1 Digits of π



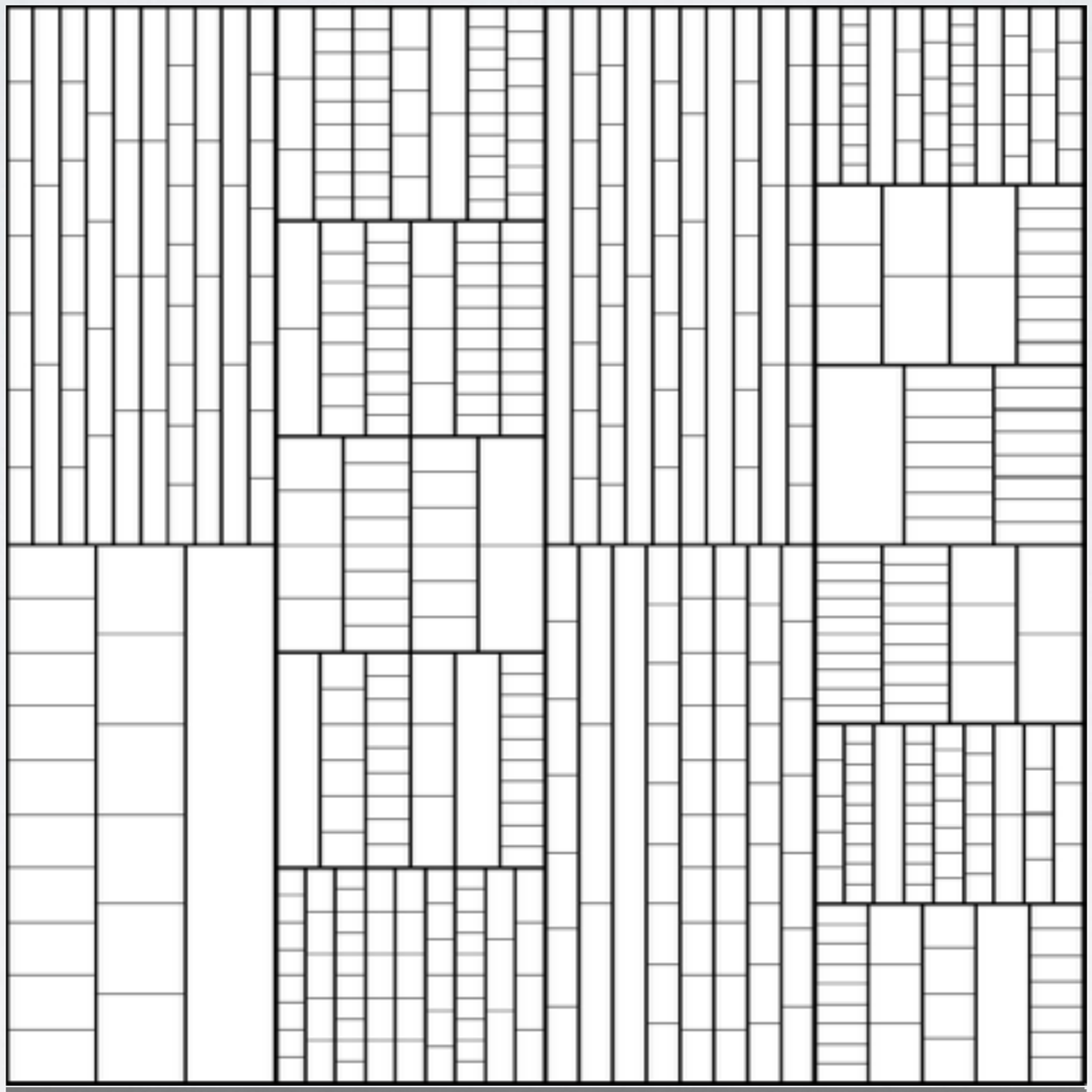
5 Digits of π



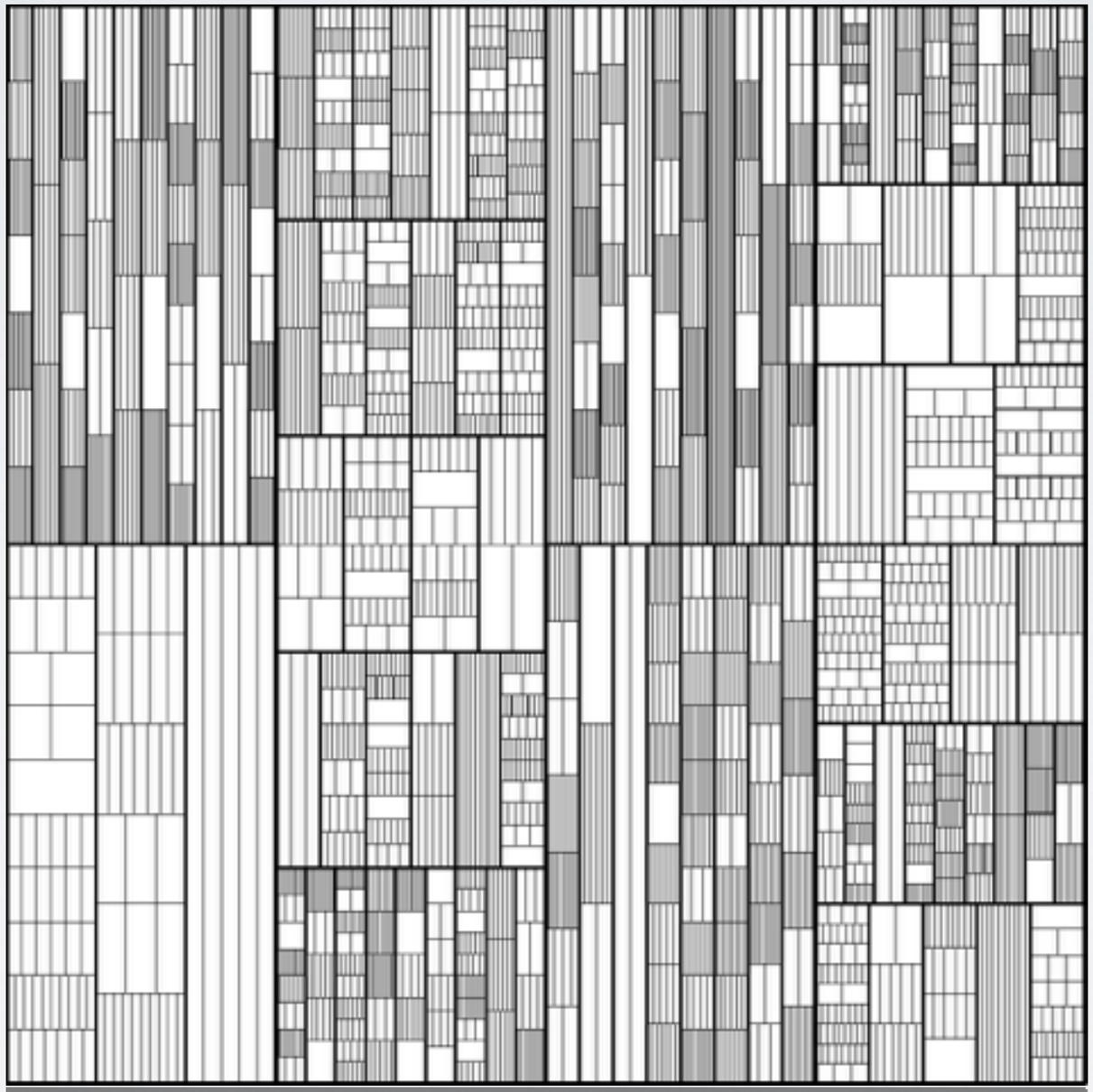
20 Digits of π



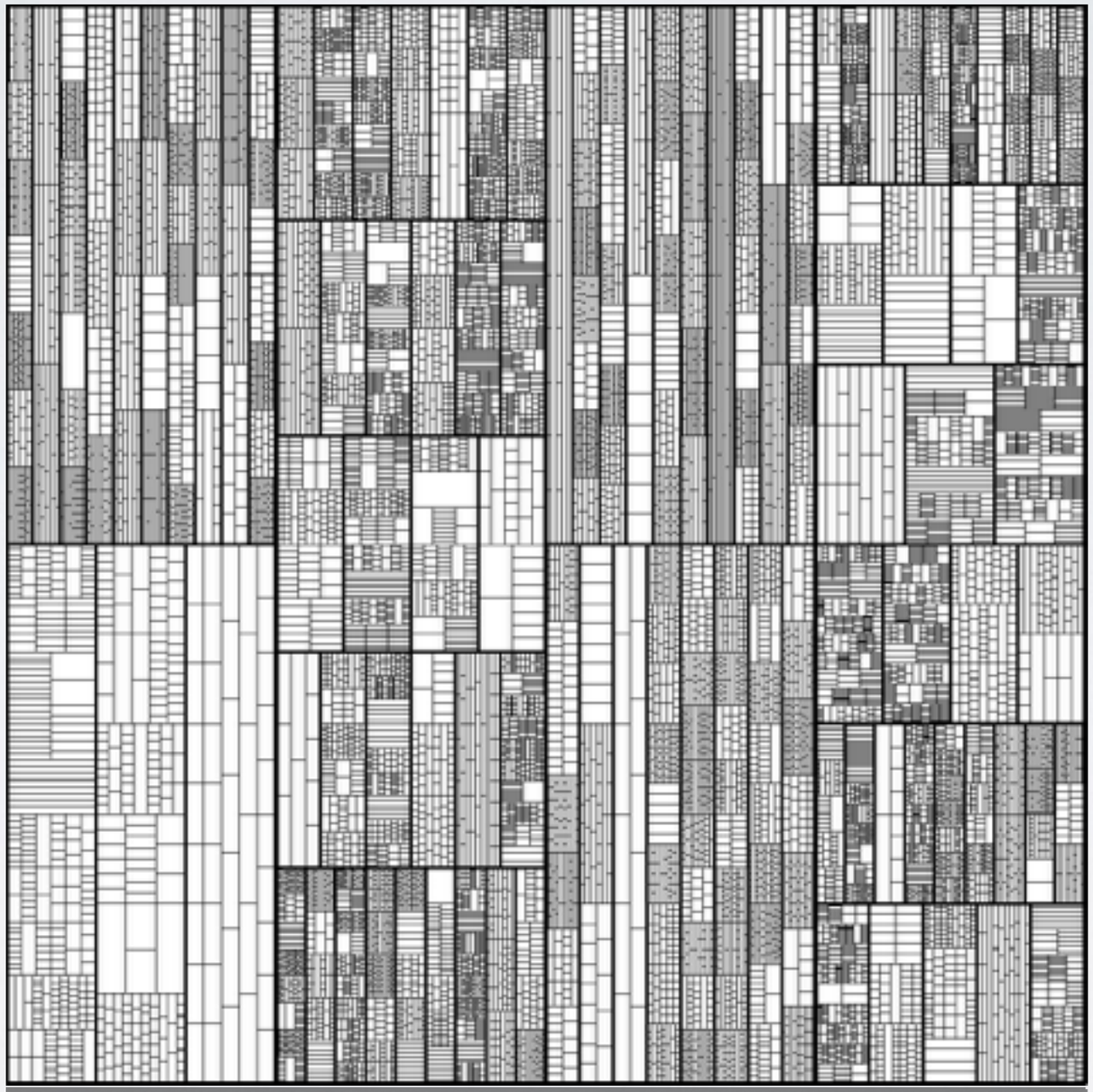
118 Digits of π



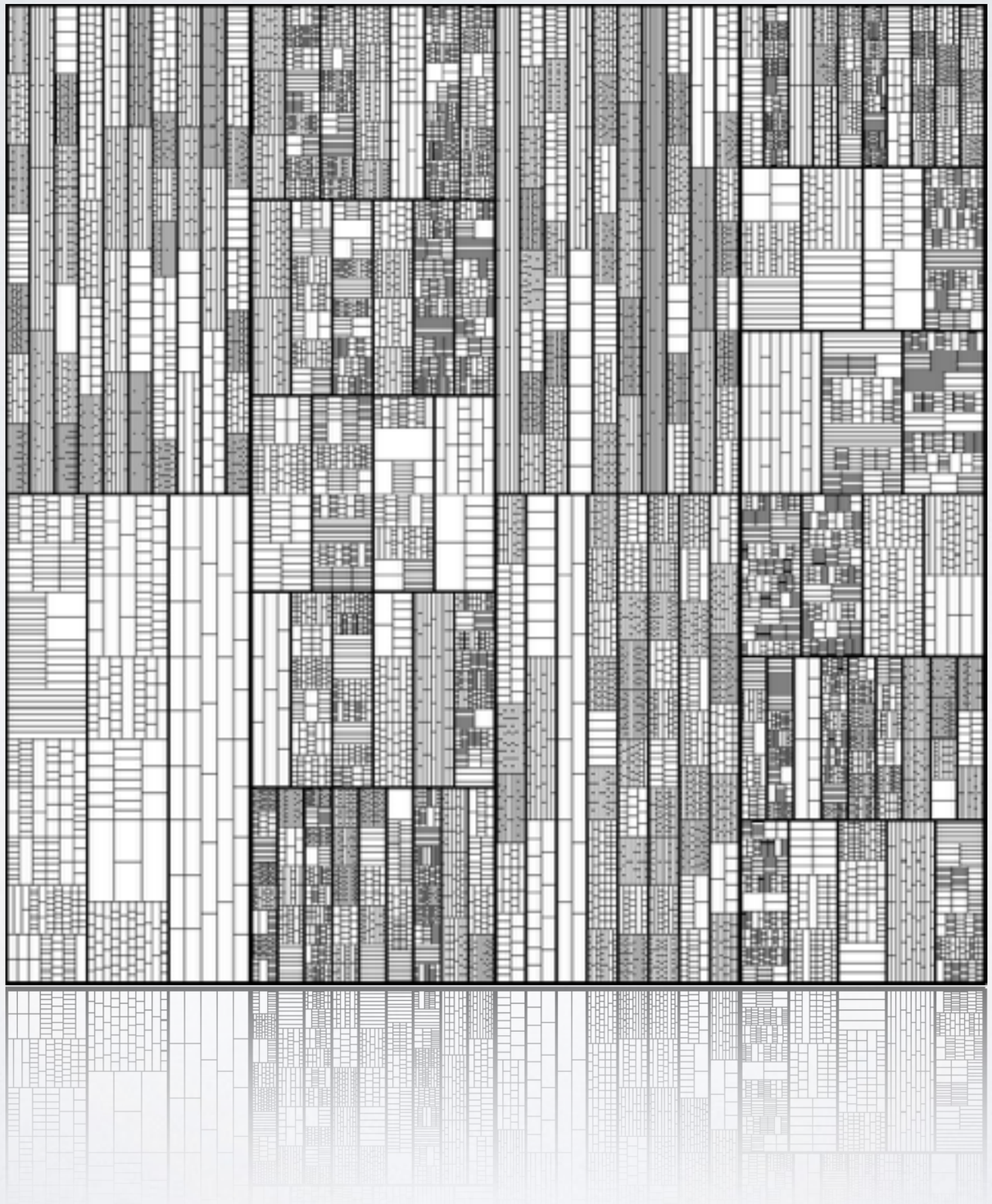
666 Digits of π



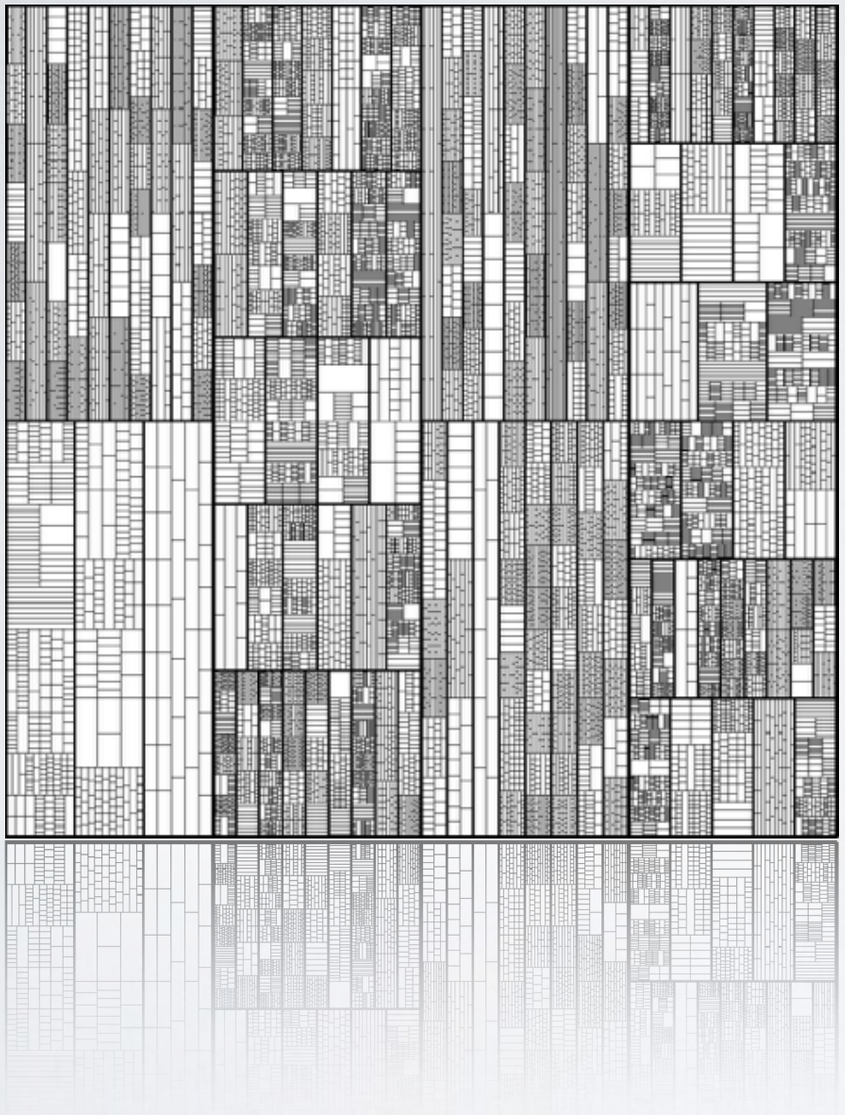
3628 Digits of π



HOW WOULD WE MAKE THIS IMAGE IN PYTHON?



HOW WOULD WE MAKE THIS IMAGE IN PYTHON?



- iterate through the digits of **π**
- partition the unit square into rectangles

Iterators

```
class yrange:

    def __init__(self, start, finish):
        self._current = start
        self._finish = finish

    def __iter__(self):
        return self

    def __next__(self):
        if self._current < self._finish:
            y = self._current
            self._current += 1
            return y
        else:
            raise StopIteration()
```

- `__next__` method is what makes an object an iterator
- Returns next thing in the iteration sequence
- When finished iterating, raises a `StopIteration` exception
- Iterators are also iterable, so they should have an `__iter__` method that returns themselves

Generators

```
class yrange:

    def __init__(self, start, finish):
        self._current = start
        self._finish = finish

    def __iter__(self):
        return self

    def __next__(self):
        if self._current < self._finish:
            y = self._current
            self._current += 1
            return y
        else:
            raise StopIteration()
```

```
def zrange(start,finish):
    while start < finish:
        yield start
        start = start + 1
```

Generating Digits of π

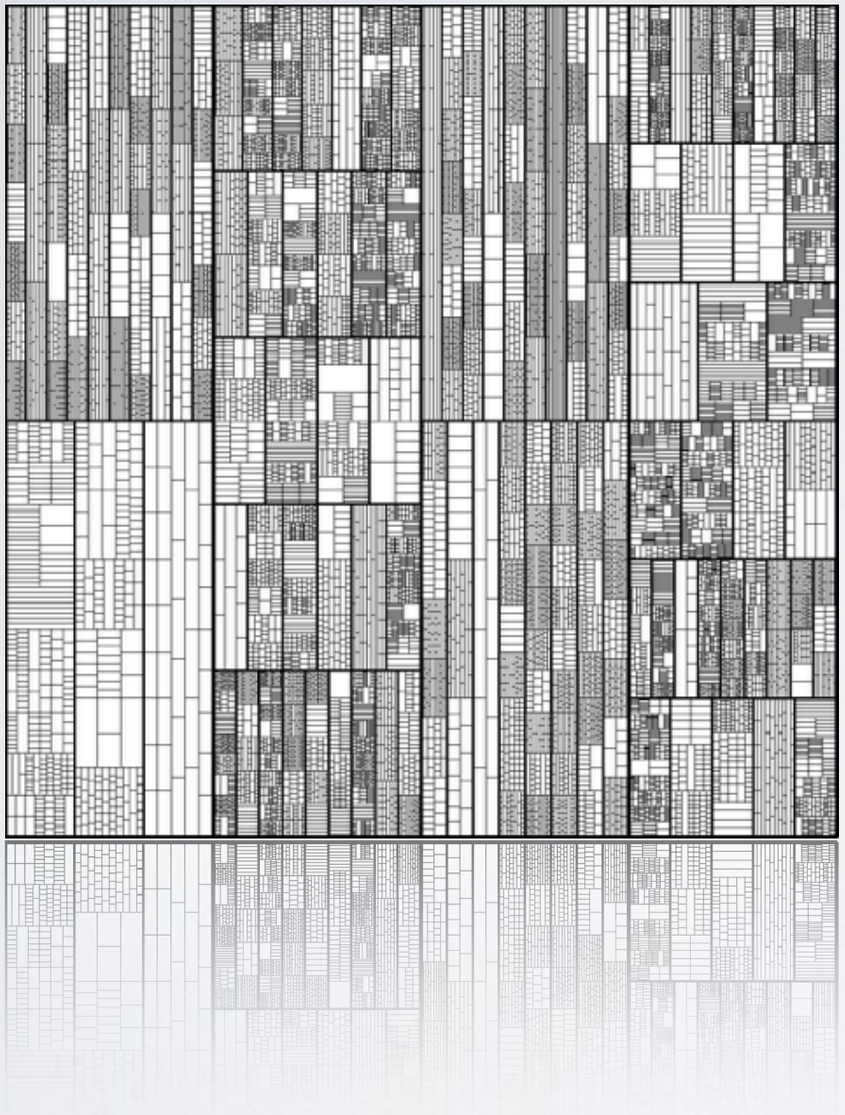
Stanley Rabinowitz and Stan Wagon
"A Spigot Algorithm for the Digits of π"
American Mathematical Monthly,102:3, March 1995, 195-203

David H. Bailey, Peter B. Borwein and Simon Plouffe
"On the Rapid Computation of Various Polylogarithmic Constants,"
Mathematics of Computation, vol. 66, no. 218 (Apr 1997), pg. 903–913.

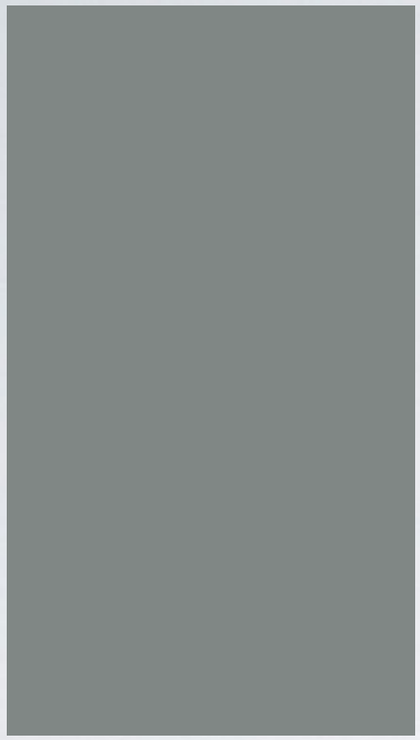
```
Program Pi_Spigot;
const n      = 1000;
len          = 10*n div 3;
var   i, j, k, q, x, nines, predigit : integer;
      a : array[1..len] of longint;
begin
  for j := 1 to len do a[j] := 2;          {Start with 2s}
  nines := 0; predigit := 0       {First predigit is a 0}
  for j := 1 to n do
  begin q := 0;
    for i := len downto 1 do              {Work backwards}
    begin
      x := 10*a[i] + q*i;
      a[i] := x mod (2*i-1);
      q := x div (2*i-1);
    end;
    a[1] := q mod 10; q := q div 10;
    if q = 9 then nines := nines + 1
    else if q = 10 then
         begin write(predigit+1);
           for k := 1 to nines do write(0);       {zeros}
           predigit := 0; nines := 0
         end
         else begin
           write(predigit); predigit := q;
           if nines <> 0 then
           begin
             for k := 1 to nines do write(9);
             nines := 0
           end
         end
  end;
  writeln(predigit);
end.
```

```
def _pi_spigot(n):
     length = (10*n)//3
      a = [2]*length
      nines = 0
      predigit = 0
      for j in range(n):
          q = 0
          for i in range(length-1,-1,-1):
              x = 10*a[i] + q*(i+1)
              a[i] = x % (2*(i+1)-1)
              q = x // (2*(i+1)-1)
          a[0] = q % 10
          q = q // 10
          if q == 9:
              nines = nines + 1
          elif q == 10:
              yield predigit+1
              for k in range(nines):
                  yield 0
              predigit = 0
              nines = 0
          else:
              yield predigit
              predigit = q
              if nines != 0:
                  for k in range(nines):
                      yield 9
                  nines = 0
yield predigit
```

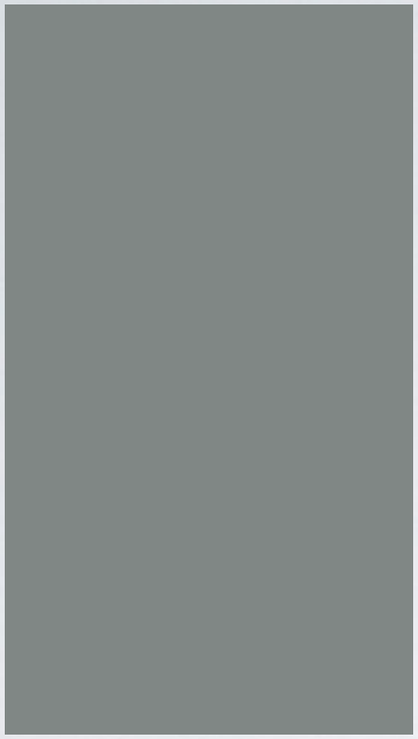
HOW WOULD WE MAKE THIS IMAGE IN PYTHON?



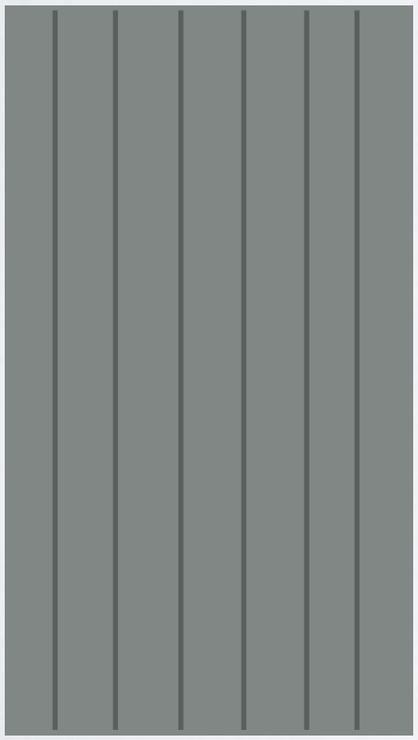
- iterate through the digits of **π**
- partition the unit square into rectangles (use an iterator)

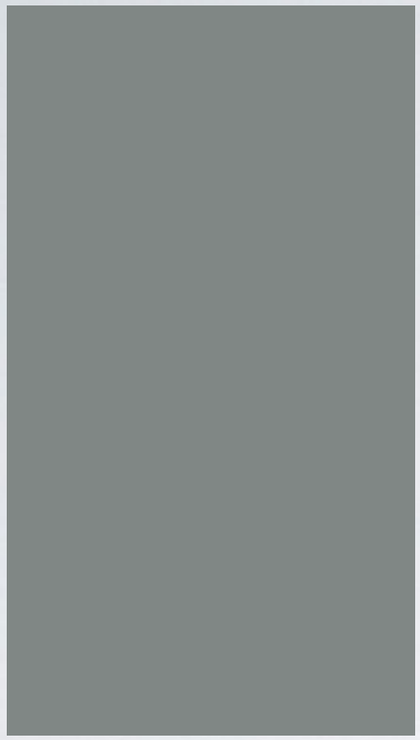


\+ VERTICAL or HORIZONTAL + Digit of **π**

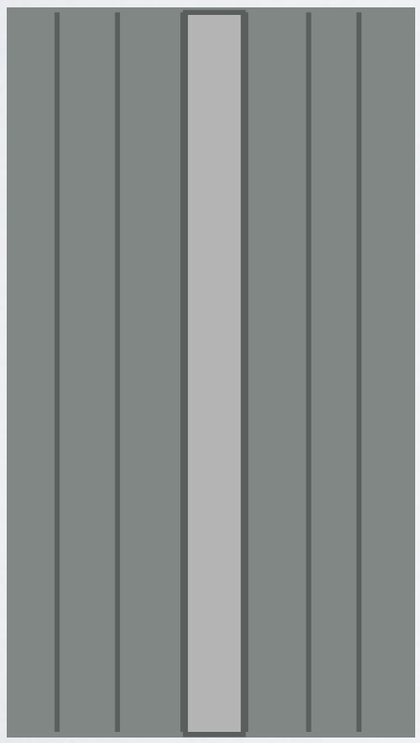


\+ **VERTICAL** or HORIZONTAL + Digit of 3.1415**5**92



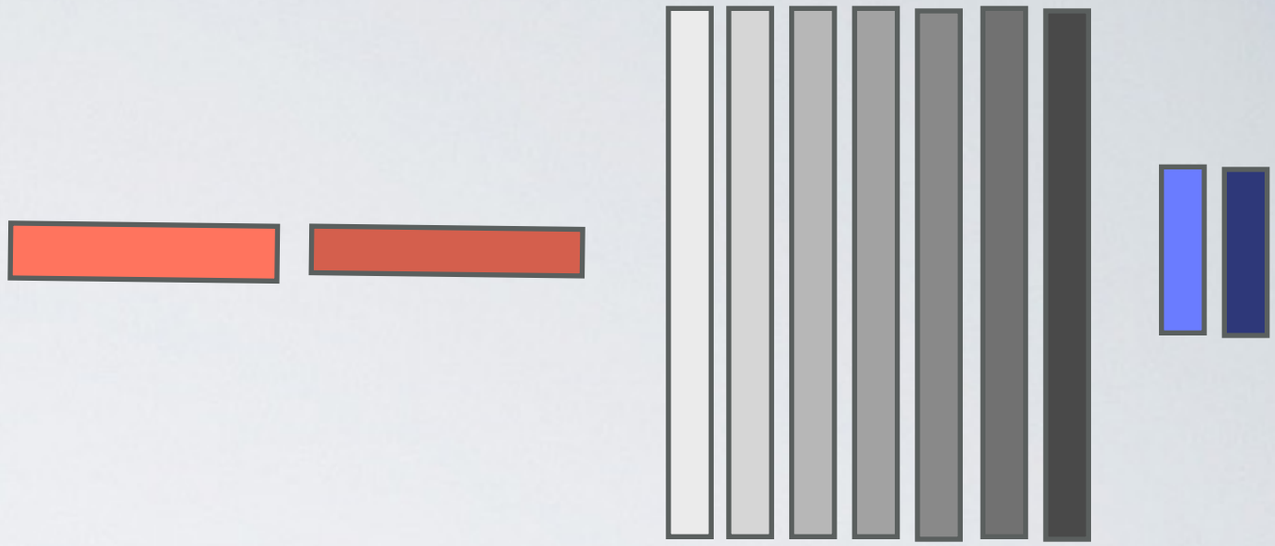


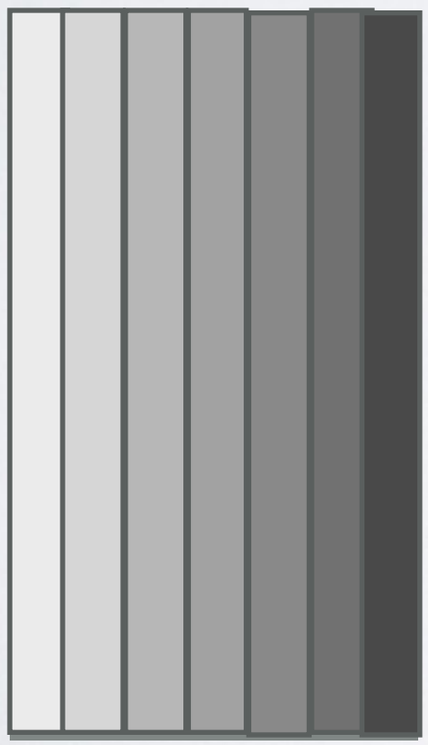
+ **VERTICAL** or HORIZONTAL + Digit of 3.1415**9**2



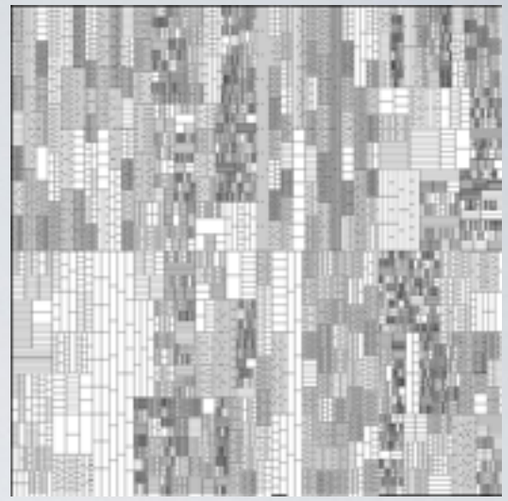
+ HORIZONTAL + Next digit of **π** given by iterator

Queue: a list of objects
Process: first in first out (fifo)





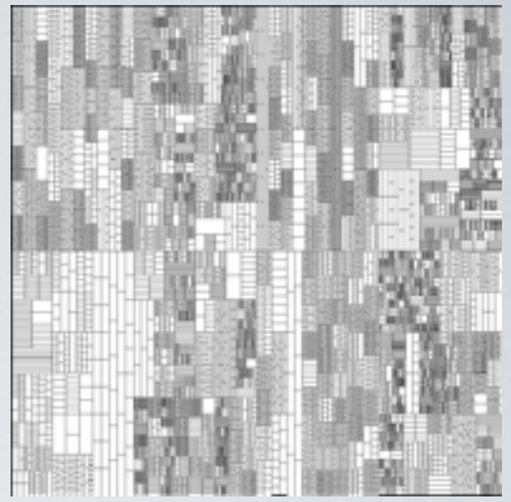
\+ HORIZONTAL \+ Next digit of **π** given by iterator



(1) Grab a (rectangle, direction, digit) triple from our queue and yield it

(2) Partition it according to the direction

(3) Add the new rectangles to the queue with the opposite direction and a new digit of **π**



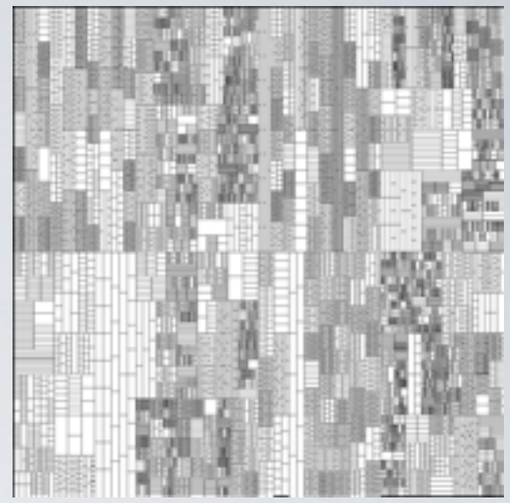
(0) WHILE (digit count > 0)

(1) Grab a (rectangle, direction, digit) triple from our queue and yield it

(2) Partition it according to the direction

(3) Add the new rectangles to the queue with the opposite direction and a new digit of **π**

(4) decrement the digit count



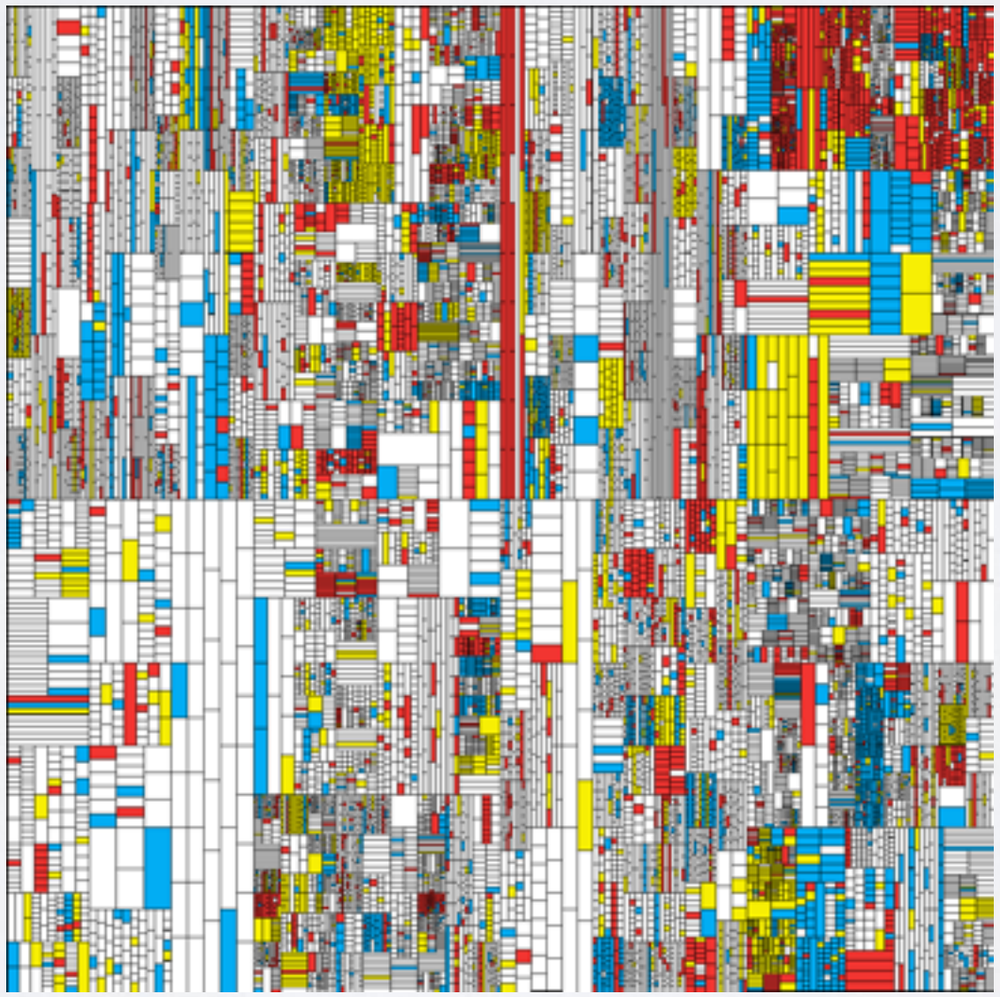
(0) WHILE (digit count > 0)

- (1) Grab a (rectangle, direction, digit) triple from our queue and yield it
- (2) Partition it according to the direction
- (3) Add the new rectangles to the queue with the opposite direction and a new digit of **π**
- (4) decrement the digit count

(5) yield the remaining rectangles in the queue

ADDING COLOR

20:1:1:1:1
Transparent : Blue : Red : Yellow : White



ADDING COLOR

20:1:1:1:1
Transparent : Blue : Red : Yellow : White

```
def rcolor(rng):
r = rng.randrange(1,24)
if r <= 20:
  return None
elif r == 21:
  return (2,173,243)
elif r == 22:
  return (242,53,49)
elif r == 23:
  return(247,239,4)
else:
  return(255,255,255)
```

ADDING LEVELS

20:1:1:1:1
Transparent : Blue : Red : Yellow : White