
Visual Arts with Python Documentation

Release 3.6

Agiliq and Contributors

Jun 04, 2018

Chapters:

1	Introductions	3
1.1	How to read this book	3
1.2	Who is this book for	3
2	Paint with prime numbers	5
2.1	A prime number which looks like salvador dali	5
2.2	A prime number whose binary representation looks like a giraffe	9
3	Generating patterns	11
3.1	Particles	11
3.2	Triangles	11
3.3	Inspirations	11
4	Gravatar, robohash and a methods for count(UUID) images.	13
4.1	Looking at robohash	13
4.2	A image for every UUID	13
5	Manipulating images	15
6	Indices and tables	17

Let us make some beautiful things with Python.

CHAPTER 1

Introductions

1.1 How to read this book

1.2 Who is this book for

Paint with prime numbers

2.1 A prime number which looks like salvador dali



Fig. 1: number theorist

Prime numbers are ubiquitous. Prime numbers are fun. Prime numbers have many uses.

But the most important use of prime numners is in art. If you haven't read [prime numbers which look like giraffe](#), you should read it, but today we are doing something more fun. Finding prime numbers which look like famous artists.

We will use Salvador Dali, my favorite surrealist, as the guinea pig.

We will use the below striking image of Dali as our source image.



This is too large (on my machine) to parse as a number, so we will resize it to 24x24 grayscale image. It looks like this



Fig. 2: Salvador Dali

Much smaller but still recognizable as Salvador. Lets get to work.

```
array([[1, 1, 1, 1, 4, 6, 7, 7, 8, 8, 7, 3, 1, 1, 6, 9, 9, 9, 9],
       [1, 1, 1, 2, 4, 5, 7, 8, 8, 8, 8, 8, 5, 1, 5, 9, 9, 9, 9],
       [1, 1, 1, 2, 3, 5, 7, 8, 8, 8, 8, 8, 8, 6, 3, 9, 9, 9, 9],
       [1, 1, 2, 3, 4, 5, 7, 8, 8, 8, 8, 8, 8, 7, 2, 8, 9, 9, 9],
       [1, 1, 2, 4, 4, 4, 4, 4, 7, 8, 8, 8, 7, 7, 3, 6, 9, 9, 9],
       [1, 1, 3, 4, 2, 2, 3, 4, 2, 4, 7, 8, 7, 6, 2, 4, 9, 9, 9],
       [1, 2, 4, 5, 3, 2, 2, 4, 4, 2, 4, 7, 7, 6, 4, 7, 9, 9, 9],
       [2, 2, 4, 5, 7, 5, 2, 5, 3, 4, 6, 4, 2, 2, 7, 9, 9, 9, 9],
       [2, 1, 2, 4, 7, 8, 5, 5, 7, 7, 7, 1, 2, 2, 7, 9, 9, 9, 9],
       [2, 2, 2, 4, 5, 7, 7, 7, 7, 7, 7, 4, 4, 6, 9, 9, 9, 9, 9],
       [2, 2, 2, 4, 4, 7, 8, 7, 6, 7, 7, 4, 5, 9, 9, 9, 9, 9, 9],
       [2, 2, 2, 4, 5, 5, 7, 4, 5, 7, 7, 4, 7, 9, 9, 9, 9, 9, 9],
       [2, 3, 3, 3, 4, 4, 7, 4, 1, 4, 5, 5, 8, 9, 9, 8, 9, 9, 9],
       [1, 2, 4, 5, 5, 2, 2, 5, 2, 1, 2, 6, 9, 9, 8, 8, 9, 9, 9],
       [1, 2, 3, 5, 5, 4, 2, 2, 3, 1, 4, 8, 8, 8, 9, 9, 9, 9, 9],
       [1, 1, 2, 4, 5, 4, 4, 4, 2, 1, 5, 8, 8, 9, 9, 9, 9, 9, 9],
       [1, 1, 1, 2, 5, 6, 4, 3, 2, 4, 9, 9, 9, 9, 9, 9, 9, 9, 9],
       [1, 1, 1, 1, 2, 5, 6, 5, 3, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9],
       [1, 1, 1, 1, 1, 2, 2, 4, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
       [1, 1, 1, 4, 7, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
       [1, 1, 4, 9, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
```

(continues on next page)

(continued from previous page)

```
[1, 1, 4, 7, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9],
[1, 1, 1, 4, 8, 8, 8, 9, 8, 8, 8, 9, 9, 9, 9, 9, 9],
[1, 1, 1, 6, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9]])
```

```
111146778873116999911124578888851599991112357888888639999112345788888872899911244444788877369991134223424787
111146778873116999911124578888851599991112357888888639999112345788888872899911244444788877369991134223424787
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10e8cc978>
```

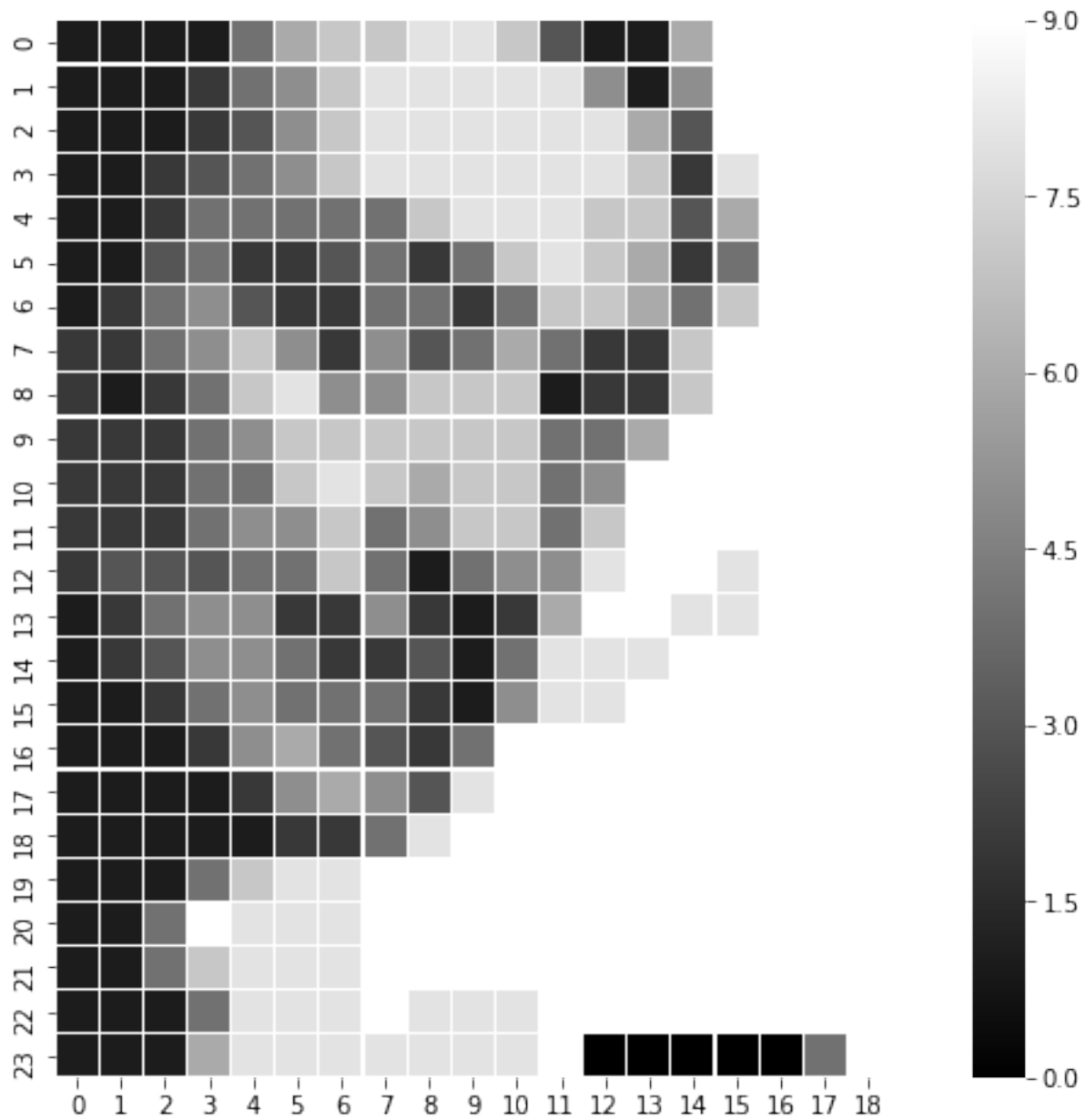
Slightly surreal, but still distinctly Salvador Dali. We have found the prime number

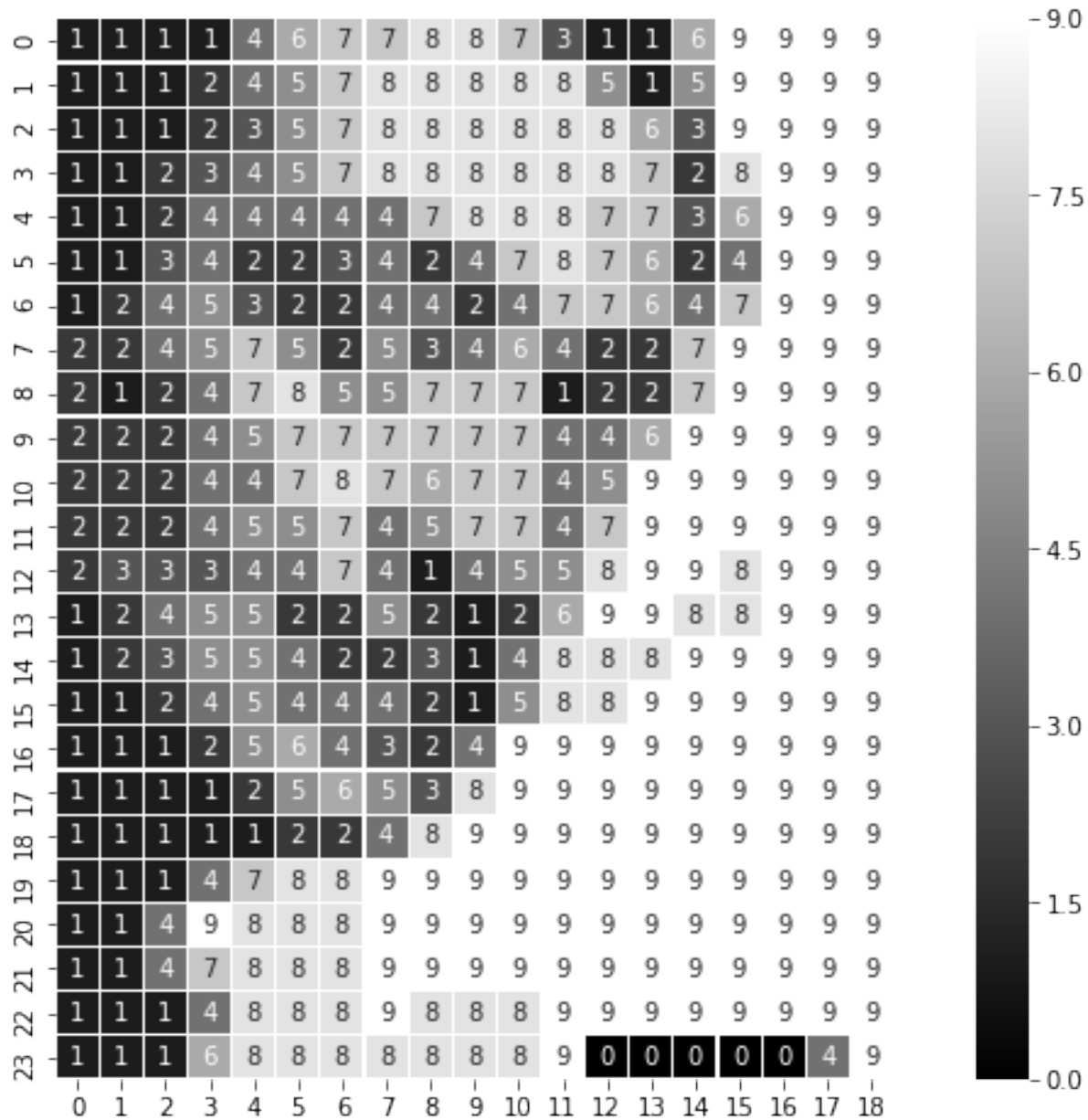
```
111146778873116999911124578888851599991112357888888639999112345788888872899911244444788877369991134223424787
```

Which when plotted in 2d looks like our Favorite Artist.

```
Populating the interactive namespace from numpy and matplotlib
```

```
/Users/shabda/.virtualenvs/yanny_vs_laurel/lib/python3.6/site-packages/
↳ IPython/core/magics/pylab.py:160: UserWarning: pylab import has clobbered
↳ these variables: ['number']
%matplotlib prevents importing * from pylab and numpy
"n`%matplotlib` prevents importing * from pylab and numpy"
```





2.2 A prime number whose binary representation looks like a giraffe

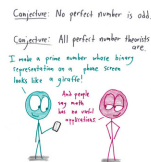


Fig. 3: Odd Number Theorists

First, here is a prime number whose binary representation is a T-Rex.

Math with bad drawings [asked](#) for a prime number whose binary representation is a giraffe. This led to discussion on [Math.reddit](#) and [Hacker news](#) which led to finding such a prime number.

```
34258179264912767619812779140611964405485280918475051120477099221060182593838317322862536861251234352
```

You can see the binary representation on reddit.

As Nathaniel Borenstein [said](#) It should be noted that no ethically-trained software engineer would ever consent to write a DestroyBaghdad procedure. Basic professional ethics would instead require him to write a DestroyCity procedure, to which Baghdad could be given as a parameter.

So the obvious next step is to generalize this to program which can take an image and find a primary number whose binary representation is the image.

We run this for an Argentinosaurus image

```
python prime_dinosaur.py -f ~/Downloads/argentinosaurus.jpg -s 40
```

Which gives us

2.2.1 How does this work?

- We read the image and convert to desired size
- The image data is converted to monochrome and pixels darker (lower) than a threshold are converted 1, rest pixels are zero.
- This data is read in in a numpy array
- This 2d array is flattened, and treated as a bitarray to get a number
- We then start incrementing the number until we get a prime.
- The primality is tested using the Miller Rabin test.
- When such a number is found

3.1 Particles

3.2 Triangles

3.3 Inspirations

- <https://trianglify.io/>
- <https://vincentgarreau.com/particles.js/>

Gravatar, robohash and a methods for count(UUID) images.

4.1 Looking at robohash

4.2 A image for every UUID

CHAPTER 5

Manipulating images

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`