Kernel: SageMath 10.4

Prime Explorations

MATH 242 Modern Computational Mathematics

Sieve of Eratosthenes

Here is an incomplete implementation of the sieve of Eratosthenes, from the end of the last class session.

In [1]:
```
# Sieve of Eratosthenes
def sieveEratos(nMax):
    # initialize a list of numbers
    nums = list(range(2, nMax+1))

    # initalize index
    i = 0

    # main loop
    while nums[i] <= sqrt(nMax):
        # check whether we have reached the next nonzero number
        if nums[i] > 0:
            # replace all multiples of p=nums[i] with zero
            j = i + nums[i] # first position to replace with zero
            while j < len(nums):
                nums[j] = 0
                j += nums[i]  # increment j by p=nums[i]
        # increment i
        i += 1

    return [n for n in nums if n != 0] # use a list comprehension to
select nonzero values
```

In [2]:
```
primesList = sieveEratos(200)
print(primesList)
```

Out[2]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139,
149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199]
```

In [3]:
```
testList=[2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19, 0]
newList=[]
for num in testList:
    if num != 0:
        newList.append(num)
newList
```

Out[3]:
```
[2, 3, 5, 7, 11, 13, 17, 19]
```

In [4]:
```
testList = [2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19,
0]
[num for num in testList if num != 0]
```

Out[4]:
```
[2, 3, 5, 7, 11, 13, 17, 19]
```

How efficient is the sieve of Eratosthenes?

In [5]:
```
%time primesList = sieveEratos(1000000)
```

Out[5]:
```
CPU times: user 2.55 s, sys: 67.8 ms, total: 2.62 s
Wall time: 3.44 s
```

In [6]:
```
import time
startTime = time.time()
primesList = sieveEratos(1000000)
endTime = time.time()
timeElapsed = endTime - startTime
```

In [7]:
```
timeElapsed
```

Out[7]:
```
5.48017144203186
```

In [8]:
```
def measureRuntime(nMax):
    startTime = time.time()
    primesList = sieveEratos(nMax)
    endTime = time.time()
    return endTime - startTime
```

In [9]:
```
measureRuntime(100000)
```

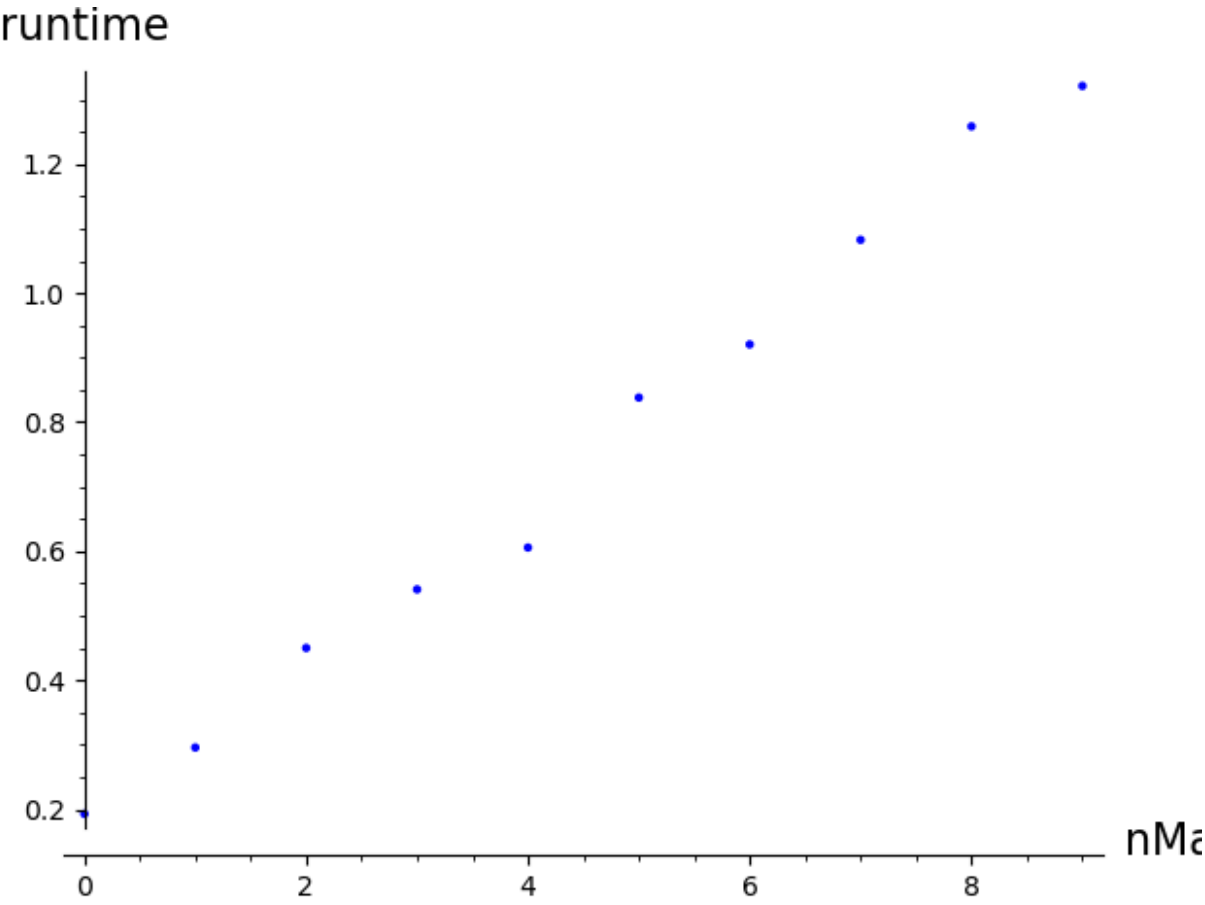
Out[9]:
```
0.42839908599853516
```

In [10]:
```
nMaxVals = range(50000, 500001, 50000) # start, stop, step
runTimes = [measureRuntime(n) for n in nMaxVals]
runTimes
```

Out[10]:
```
[0.19278621673583984,
 0.29561591148376465,
 0.4499237537384033,
 0.5407638549804688,
 0.605597972869873,
 0.8380863666534424,
 0.9205069541931152,
 1.0826542377471924,
 1.2584559917449951,
 1.3212025165557861]
```

In [13]:
```
list_plot(runTimes, axes_labels=['nMax', 'runtime'])
```

Out[13]:



Units Digits of Primes

The only primes with a units digit of 2 or 5 are the primes 2 and 5. All other primes have a units digit of 1, 3, 7, or 9. How often do these digits occur?

Warm-Up

Consider the primes less than 100. Of these primes, count how many have each units digit 1, 3, 7, and 9.

In [12]:

In [0]:

Exploration

Now replace 100 by some other integer M. How many primes less than or equal to M have each units digit 1, 3, 7, and 9? Consider various values of M.

- What patterns do you observe in your counts?
- What questions arise during your exploration?
- What conjectures can you make?

In [0]:

In [0]: