Kernel: SageMath 10.4

Counting Primes

Math 242 Modern Computational Mathematics

First, here is a copy of our sieve of Eratosthenes from the previous class.

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
```

The Prime Counting Function

Define the **prime counting function** $\pi(x)$ to be the number of primes less than or equal to any real number x. For example, $\pi(10) = 4$ since there are 4 primes less than or equal to 10: specifically, 2, 3, 5, and 7.

Perhaps the simplest way to compute $\pi(x)$ is to find the length of the list returned by `sieveEratos(x)` .

In [2]:
```
def pi(x):
    return len(sieveEratos(x))
```

For example:

In [6]:
```
pi(1000)
```

Out[6]: 168

In [5]:
```
print(sieveEratos(100))
```

Out[5]:
```
[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]
```

We can then plot values of the prime counting function. First we need to make a list of values of $\pi(x)$.

In [7]:
```
nMax = 100
piVals = [pi(n) for n in range(2, nMax)]
print(piVals)
```

Out[7]:
```
[1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8, 8, 8, 8, 9, 9, 9,
9, 9, 9, 10, 10, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13, 14, 14,
14, 14, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 17, 17, 18, 18,
18, 18, 18, 18, 19, 19, 19, 19, 20, 20, 21, 21, 21, 21, 21, 21, 22, 22,
22, 22, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 25, 25,
25]
```

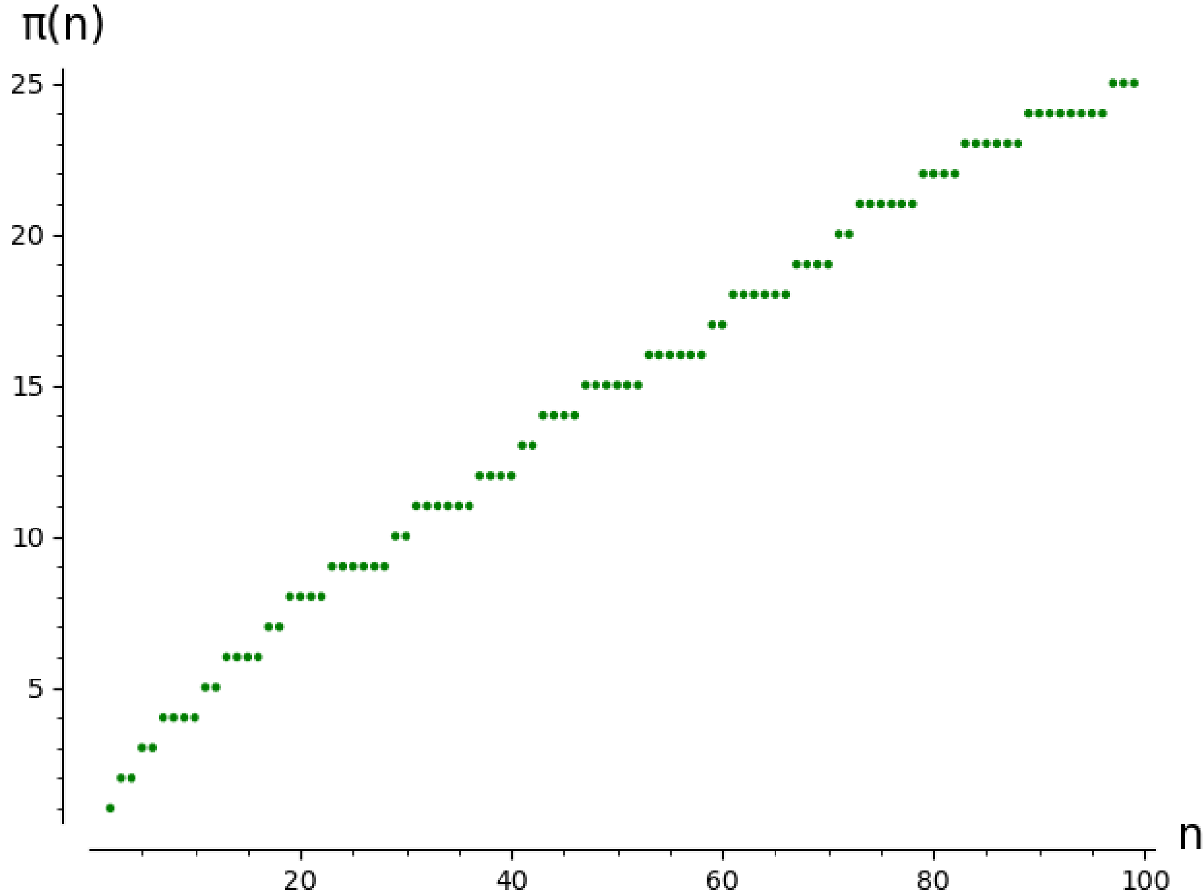
Now we can plot the prime counting function.

In [13]:
```
print( list( zip( range(2,nMax), piVals ) ) )
```

Out[13]:
```
[(2, 1), (3, 2), (4, 2), (5, 3), (6, 3), (7, 4), (8, 4), (9, 4), (10,
4), (11, 5), (12, 5), (13, 6), (14, 6), (15, 6), (16, 6), (17, 7), (18,
7), (19, 8), (20, 8), (21, 8), (22, 8), (23, 9), (24, 9), (25, 9), (26,
9), (27, 9), (28, 9), (29, 10), (30, 10), (31, 11), (32, 11), (33, 11),
(34, 11), (35, 11), (36, 11), (37, 12), (38, 12), (39, 12), (40, 12),
(41, 13), (42, 13), (43, 14), (44, 14), (45, 14), (46, 14), (47, 15),
(48, 15), (49, 15), (50, 15), (51, 15), (52, 15), (53, 16), (54, 16),
(55, 16), (56, 16), (57, 16), (58, 16), (59, 17), (60, 17), (61, 18),
(62, 18), (63, 18), (64, 18), (65, 18), (66, 18), (67, 19), (68, 19),
(69, 19), (70, 19), (71, 20), (72, 20), (73, 21), (74, 21), (75, 21),
(76, 21), (77, 21), (78, 21), (79, 22), (80, 22), (81, 22), (82, 22),
(83, 23), (84, 23), (85, 23), (86, 23), (87, 23), (88, 23), (89, 24),
(90, 24), (91, 24), (92, 24), (93, 24), (94, 24), (95, 24), (96, 24),
(97, 25), (98, 25), (99, 25)]
```

In [21]:
```
list_plot( list( zip( range(2,nMax), piVals ) ), axes_labels=
["n","π(n)"], color="green" )
```

Out[21]:



In [18]:

```
α = 5
```

In [19]:

```
α^2
```

Out[19]: 25

In [23]:

```
nMax = 10000
piVals = [pi(n) for n in range(2, nMax)]
list_plot( list( zip( range(2,nMax), piVals ) ), axes_labels=
["n","π(n)"], color="green" )
```

Expand

Unfortunately, this is inefficient because we are running the sieve of Eratosthenes for each individual data point above.

We should be able to compute a single list of primes up to N and get all of the counts from that list. Let's do that in the next code cell:

In [24]:

```
[0]*20
```

Out[24]: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

In [27]:

```
# returns a list of values of the prime-counting function π(x), for
integers x from 1 to nMax
def computePiVals(nMax):
```

```
    # compute a list of primes up to nMax
    primeList = sieveEratos(nMax)

    # make a list of nMax+1 zeros
    piVals = [0]*(nMax+1)

    # track how many primes we've found so far
    count = 0

    # loop over integers i from 2 to nMax
    for i in range(2, nMax + 1):
        # if i is the next prime, then add 1 to our count
        if count < len(primeList) and i == primeList[count]:
            count += 1  # we found the next prime

        # store the current count in piVals[i]
        piVals[i] = count

    # return the list of piVals
    return piVals
```

Try out this new function:

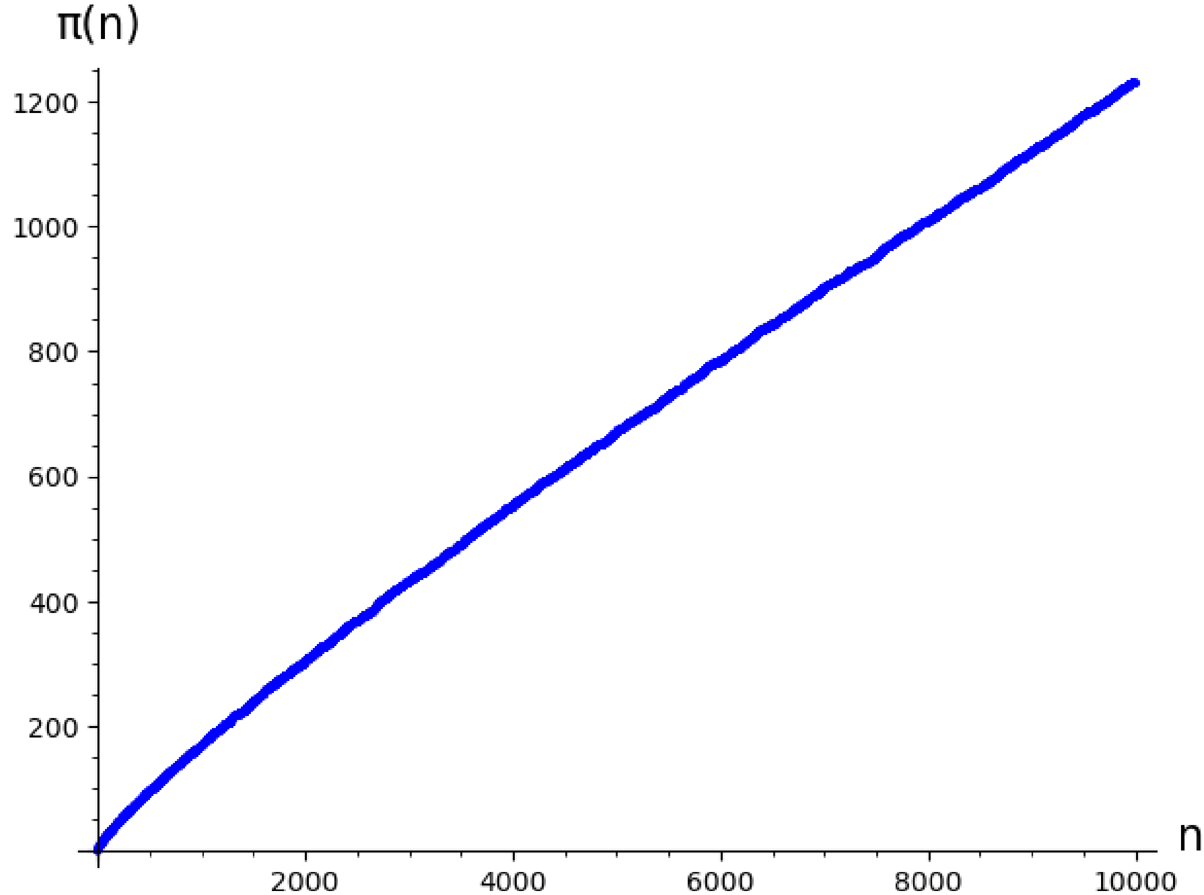
In [29]:
```
print(computePiVals(100))
```

Out[29]:
```
[0, 0, 1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8, 8, 8, 8, 9,
9, 9, 9, 9, 9, 10, 10, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 17, 17,
18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 20, 20, 21, 21, 21, 21, 21, 21,
22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25]
```

Now we can quickly compute a huge list of values of $\pi(x)$ and make a plot.

In [31]:
```
nMax = 10000
piVals = computePiVals(nMax)
list_plot(piVals, axes_labels=["n","π(n)"])
```

Out[31]:



Exploration

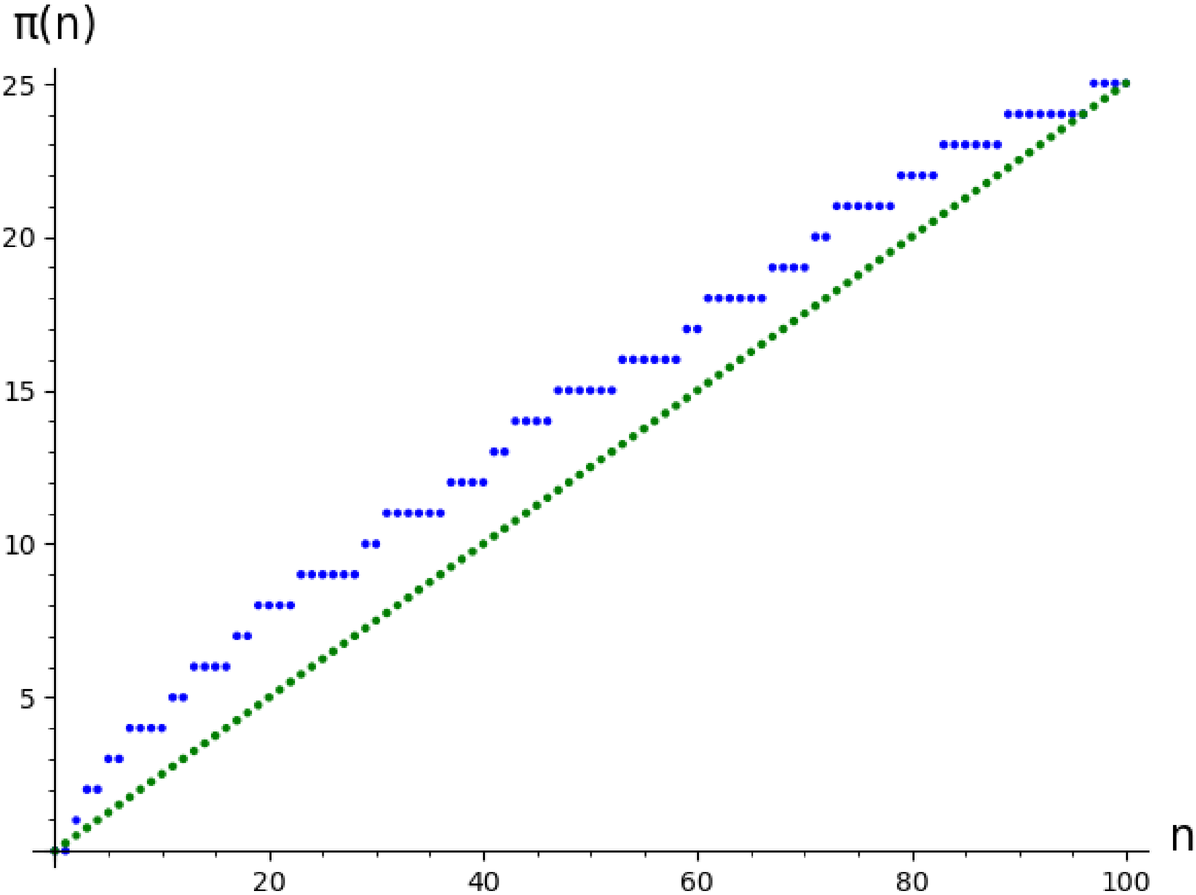
Discuss with your group the following questions:

1. What is the shape of the graph of $\pi(x)$? Can you find a simple function that approximates $\pi(x)$?
2. What proportion of the first N positive integers are prime? How does this depend on N?
3. Use your best answers to the previous questions to the previous questions, how many primes do you think are less than 10^{20}? How many primes do you think are less than 10^{100}?

In [34]:

```
nMax = 100
piVals = computePiVals(nMax)
linearVals = [n/4 for n in range(nMax+1)]
list_plot(piVals, axes_labels=["n","π(n)"]) + list_plot(linearVals,
color="green")
```

Out[34]:



In [0]: