Kernel: SageMath 10.7

Counting Primes

MATH 242 Modern Computational Mathematics

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

In [19]:
```
# function: sieveEratos
# input: integer nMax, which must be at least 2
# output: a list of primes up to nMax
def sieveEratos(nMax):
    # create a list of integers up to nMax
    # if you start the list from zero, then each number in the list is
EQUAL to its index in the list, which is very convenient
    nums = list(range(nMax+1))

    # replace 1 in the list with 0
    nums[1] = 0

    # compute sqrt(nMax)
    nroot = floor(sqrt(nMax))

    # loop over each list item less than or equal to nroot
    i = 0
    while nums[i] <= nroot:
        # if nums[i] is not zero, then i is prime
        if nums[i] != 0:
            # i is prime, so set all of its multiples to zero
            j = i^2
            while j <= nMax:
                nums[j] = 0
                j += i
        i += 1

    # now extract all nonzero numbers from the list
    return [n for n in nums if n != 0]

# testing
print(sieveEratos(100))
```

Out[19]:
```
[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]
```

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 [20]:
```
def pi(x):
    return len(sieveEratos(floor(x)))
```

For example:

In [3]:
```
pi(100)
```

Out[3]: 25

In [6]:
```
pi(100.4)
```

Out[6]: 25

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

In [21]:
```
nMax = 100
piVals = [pi(n) for n in range(2, nMax)]
print(piVals)
```

Out[21]:
```
[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]
```

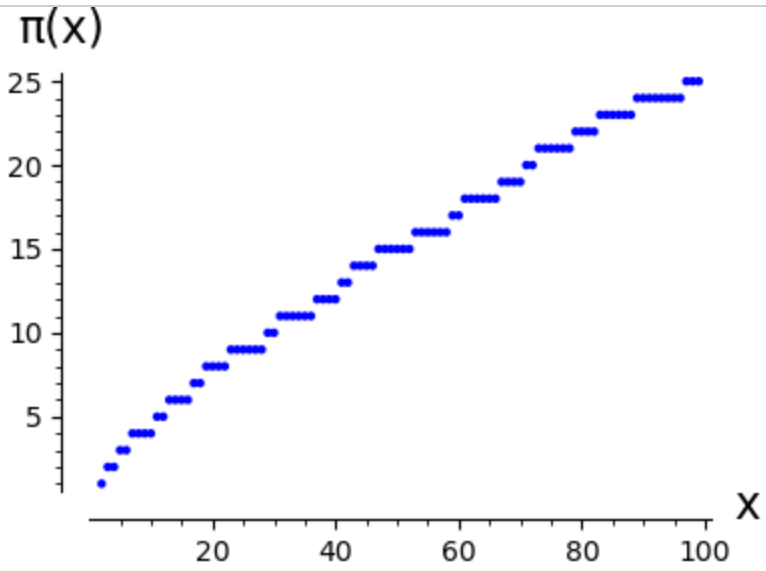
In [0]:
```

```

Now we can plot the prime counting function.

In [22]:
```
list_plot( list(zip(range(2,nMax), piVals)) , axes_labels=["x","π(x)"],
figsize=4)
```

Out[22]:



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 [15]:
```
# 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
    primes = 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(primes) and i == primes[count]:
            count += 1
        #print(f"i = {i}, count = {count}")

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


    # return the list of piVals
    return piVals
```

Try out this new function:

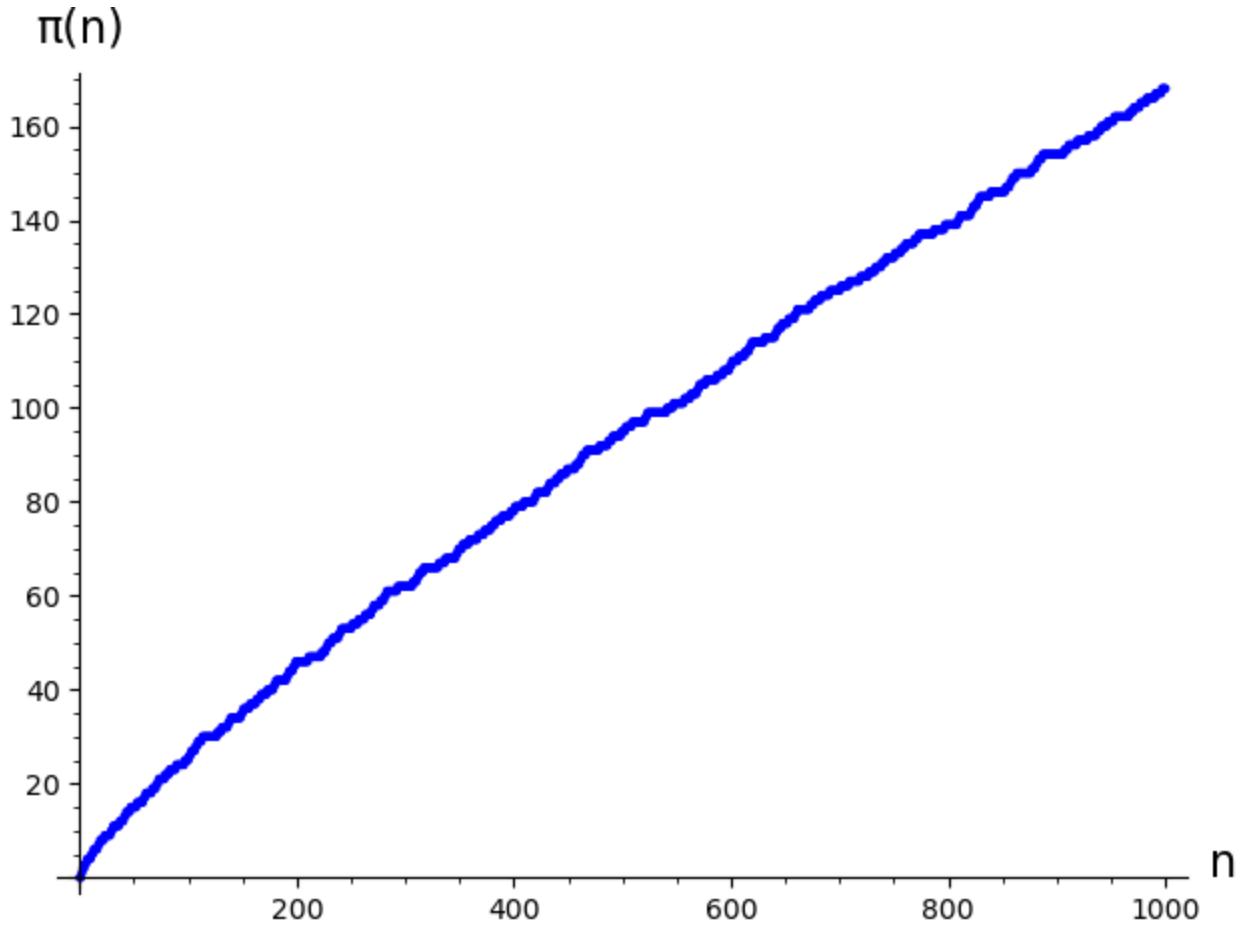
In [17]:
```
print( computePiVals(100) )
```

Out[17]:
```
[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 [18]:
```
nMax = 1000
piVals = computePiVals(nMax)
list_plot(piVals, axes_labels=["n","π(n)"])
```

Out[18]:



Exploration

Use computation to explore the following questions. Discuss your observations with your group. What can you conclude for each question?

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^{15}?
4. How many primes do you think are less than 10^{20}?
5. How many primes do you think are less than 10^{100}?

In [0]:

In [0]:

In [0]:

In [0]: