Math 414: Lecture 3

Sage Demo

Sage is:

1. A large free open source mathematics software project that I direct.
2. Web-based at http://sagenb.org or you can run it locally.
3. **Very** powerful for *number theory*, and also combinatorics, graph theory, and numerical analysis (via scipy).
4. Uses **Python** for the main user language.

GCD

gcd(2010,2007)

```
3
```

gcd([15,25,20,45])

```
5
```

gcd(90324809238420398423230948203948, 2903480923840923849082309482)

```
6
```

How to make up huge random integers:

n = ZZ.random_element(10^10000); m = ZZ.random_element(10^10000)
len(n.digits())

```
10000
```

gcd(n,m)

```
1
```

Enumerating Primes

prime_range(50)

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

prime_range(200)

```
[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]
```

@interact
def f(n=(10..2500)):

Processing Math: Done

```
print prime_range(n)
```

Sieving Primes

```
@interact
def f(n=(2..400), pmax=(2..20)):
  P = [2]
  X = [3,5,..,n]
  while True:
    if len(X) == 0: break
    p = X[0]
    if p > pmax:
      P.extend(X); break
    P.append(p)
    X = [a for a in X if a%p]
  print "<html>"
  for a in [1..n]:
    clr = '#a00' if a in P else '#aaf'
    print "<font color='%s'>%s</font>"%(clr, ' '*(3-len(str(a)))+str(a)),
    if a%20 == 0:
      print
  print "</html>"
```

Mersenne Primes

The prime below is the largest known prime. The GIMPS project found it, winning them a $100,000 prize from the EFF, since it has > ten million digits.

time p = 2^43112609 - 1

```
Time: CPU 0.01 s, Wall: 0.01 s
```

time v = p.digits()

```
Time: CPU 14.40 s, Wall: 14.49 s
```

len(v)

```
12978189
```

Last 10 digits:

v[-10:]

```
[3, 9, 6, 2, 0, 7, 4, 6, 1, 3]
```

Processing Math: Done

v[:10]

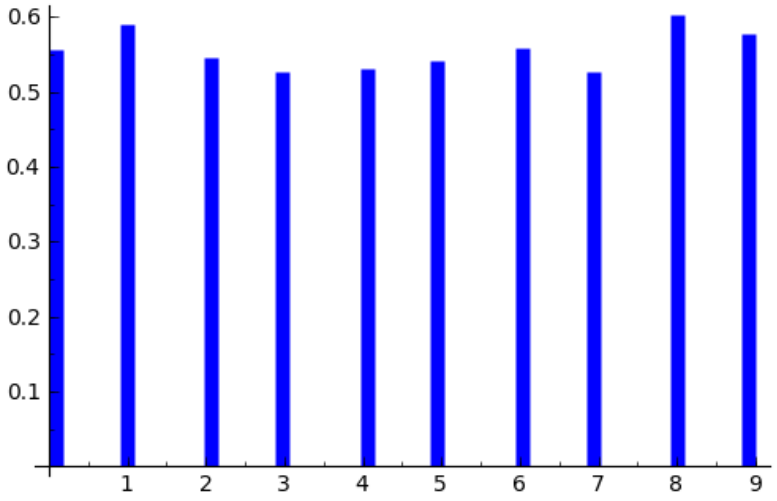
```
[1, 1, 5, 2, 5, 1, 7, 9, 6, 6]
```

Frequency histogram of first n digits:

```
@interact
def f(n=(10,100,..10^5)):
  time T = finance.TimeSeries(v[:n])
  show(T.plot_histogram())
```

n

```
Time: CPU 0.00 s, Wall: 0.00 s
```



Counting Primes

prime_pi

```
Function that counts the number of primes up to x
```

prime_pi(10)

```
4
```

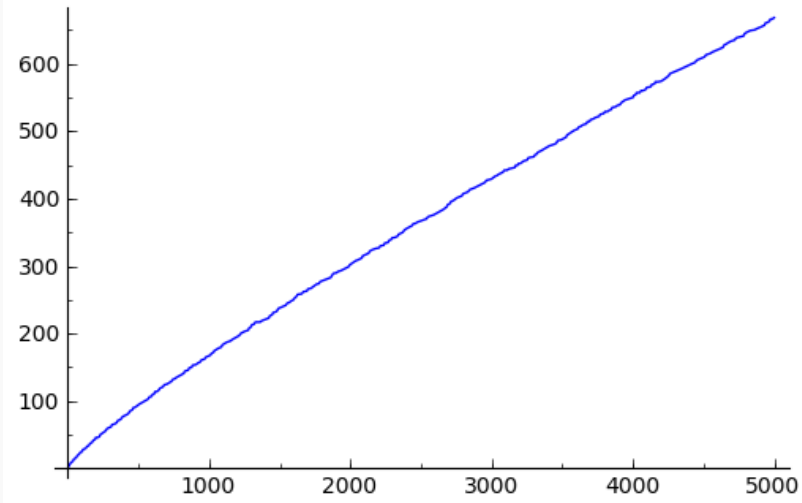
```
for n in [10,100,1000,10000,100000]:
  print n, prime_pi(n)
```

```
10 4
100 25
1000 168
10000 1229
100000 9592
```

```
@interact
def f(n=(10..5000)):
  show(plot(prime_pi, 1, n))
```

Processing Math: Done

n



Wait, that looks like a nice clean smooth curve? What is it "basically" a plot of?

The Riemann Hypothesis is a conjectural answer to this question...

Processing Math: Done