# Basics Of Sage Crypto

## BASICS OF SAGEMATH

```
var('x') # decalre a variable
```

```
x
```

```
solve(x^2 + 3*x -10.0==0.0,x) # use variable in equation and solve
```

```
[x == -5, x == 2]
```

```
del x # delete x
```

```
# Declare a variable
```

```
s=7.0
```

```
parent(s) # use parent command to check variable type
```

```
Real Field with 53 bits of precision
```

```
s = ZZ(s) # convert variable to integer ring
```

```
parent(s) # s is now an integer ring
```

```
Integer Ring
```

### Operational difference between / and //

```
# decalre another variable
b = 6/2
print(b)
```

```
3
```

```
parent(b)  # rational field type
```

```
Rational Field
```

```
# but this gives an inteher
b=6//2
```

```
parent(b)  # b is an integer ring here
```

```
Integer Ring
```

### Getting range of values

```
range(3,7,2) # get range of values between 3 and 7 , with steps 2
```

```
[3, 5]
```

```
[x^2 for x in range(2,4)] # this can be used in a for loop like this. Note no step was specified here. In the above example the step was 2
```

```
[4, 9]
```

```
[floor (sqrt(x)) for x in range(10,100,20) ] # You can also use this like this. Floor = min
```

```
[3, 5, 7, 8, 9]
```

```
[floor (x^(1/3)) for x in range(10,100,20) ] # Another exaple
```

```
[2, 3, 3, 4, 4]
```

```
[chr(x) for x in range(40,100,10)] # character input can also be returned
```

```
['(', '2', '<', 'F', 'P', 'Z']
```

## variable to a power

```
var('x') # declare variable as above
```

```
x
```

```
[x^i for i in range(1,10)] # return variable to a power
```

```
[x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9]
```

## Useful Commands

```
nth_prime(3) # return nth prime
```

```
5
```

```
q=[nth_prime(x) for x in range(1,12)] # nth prime used with the range
r = [nth_prime(x) for x in range(13,24)]
```

```
q[2:6] # view output
```

```
[5, 7, 11, 13]
```

```
primes_first_n(10) # return the first nth primes
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

```
[x+y for x,y in zip(q,r)] # note the use of zip to return p plus q
```

```
[43, 46, 52, 60, 70, 74, 84, 90, 96, 108, 114]
```

```
k=primes_first_n(12) # decalre first 12 prime
```

```
[x*y for x,y in zip(k,q)] # use zip to return multiplication
```

```
[4, 9, 25, 49, 121, 169, 289, 361, 529, 841, 961]
```

```
prime_range(11,20) # returns prime in a range specified
```

    [11, 13, 17, 19]

## GCD AND EX_GCD AND CHINESE REMAINDER THEOREM

```
gcd(25,3) # gcd output, gcd = greatest common divisor
```

    1

```
gcd(29,5) # gcd output
```

    1

```
mod(29,5) # example
```

    4

```
xgcd(45,39) # extended gcd
```

    (3, -6, 7)

```
crt(3,4,7,5) # chinese remainder
```

    24

```
mod(16,17) # get modulus
```

    16

```
mod(25,17) # get modulus
```

    8

## Power mod command

```
power_mod(83,37,191) # 83 to the power 37 mod 191
```

    58

```
power_mod(10,35,13) # 10^3 mod 13
```

    4

## MILLER RABIN TEST

```

def bmod(x,n):
    '''
    return a balanced moduli
    '''
    return (x+(n-1)//2)%n-(n-1)//2

def bseq(b,n):
    '''
    perform the miller rabin test

    '''
    s= (n-1).trailing_zero_bits();
```

```
    d = (n-1)>>s;
    return [bmod(power_mod(b,2^j*d,n),n) for j in range(s+1)]
```

```
bseq(2,3889)
```

```
[592, 454, -1, 1, 1]
```

```
3888
```