Sum of Digits

```
def sum_of_digits(n):
    # base condition
    if n == 0:
        return 0

    return n % 10 + sum_of_digits(n // 10)

sum_of_digits(124)
```

```
7
```

```
sum_of_digits(12034)
```

```
10
```

```
def sum_of_digits(n):
    # base condition
    if n == 0:
        print(n, "Base condition is hit")
        return 0
    print("Now value of n is:", n)
    return n % 10 + sum_of_digits(n // 10)

sum_of_digits(12034)
```

```
Now value of n is: 12034
Now value of n is: 1203
Now value of n is: 120
Now value of n is: 12
Now value of n is: 1
0 Base condition is hit

10
```

Power of a number

```
def power(base, pwr):
    # base condition
    if pwr == 0:
        return 1

    return base * power(base, pwr - 1)

power(2, 3)
```

```
8
```

```
power(5, 3)
```

```
125
```

```
0 ** 3
```

```
0
```

```
power(0, 3)
```

```
0
```

```
def power(base, pwr):
    # base condition
    if pwr == 0:
        return 1
    # if base is 0
    if base == 0 or base == 1:
        return base
    # Exclude -ve powers
    if pwr < 0:
        return "Invalid power"

    return base * power(base, pwr - 1)
```

```
power(0, 4)
```

```
0
```

```
power(1, 5)
```

```
1
```

```
power(2, -3)
```

```
'Invalid power'
```

Optimized power

```
# In any case I need n // 2 power

def opt_power(a, n):
    # base condition
    if n == 0:
        return 1

    half = opt_power(a, n // 2)
```

```
    # Check for even odd
    if n % 2 == 0:
        return half * half
    else:
        return a * half * half

opt_power(2, 8)
```

```
256
```

```
# Doubts

def square(n):
    return n ** 2

a = square

a(2)
```

```
4
```

```
b = square(3)

type(a)
```

```
function
```

```
type(b)
```

```
int
```