Arithmetic operators

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print (a+b)
    print (a-b)
    print (a*b)
```

division

```
if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print (a//b)
    print (a/b)
```

loop

```
if __name__ == '__main__':
    n = int(input())
    for i in range (0,n):
       print (i*i)
```

print function

```
if __name__ == '__main__':
    n = int(input())
    for i in range(n):
        print(i+1, end="")
```

```
if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores
    query_name = input()
    count = 0.00
    for i in student_marks:
        if i == query_name:
```

```
            for i in student_marks[i]:
                count+=i
    a = count/3.00
    format_float = "{:.2f}".format(a)
    print(format_float)
```

percentage finding

swap case

```
def swap_case(s):
    result = s.swapcase()
    return result

if __name__ == '__main__':
    s = input()
    result = swap_case(s)
    print(result)
```

```
#
# Complete the 'print_full_name' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. STRING first
#  2. STRING last
#

def print_full_name(first, last):
    # Write your code here
print("Hello " + first + " " + last + "! You just delved into python.")

if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_na
```

Whats your name

```
import textwrap
def wrap(string, max_width):
    wrapper = textwrap.TextWrapper(width=max_width)
    dedented_text = textwrap.dedent(text=string)
    result = wrapper.fill(text=dedented_text)
    return result

if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

text wrap

```
def mutate_string(string, position, character):
    ls = list(string)
    ls[position] = character
    return(''.join(ls))

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

mutations

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
T = int(input())
for _ in range(T):
    num_elems_A = int(input())
    set_A = set(map(int, input().split()))
    num_elems_B = int(input())
    set_B = set(map(int, input().split()))

    if set_A.issubset(set_B):
        print(True)
    else:
        print(False)
```

check subset

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
_,A,N=input(),set(map(int,input().split())),int(input())

for i in range(N):
    method,_=input().split()
    B=set(map(int,input().split()))
    getattr(A,method)(B) # for e.g. A.update(B)
print(sum(A))
```

set mutations

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
m,n=input(),set(map(int,input().split()))
a,b=input(),set(map(int,input().split()))
print(len(n|b))
```

set union

```
E = int(input())
ES = set(map(int,input().split()))
F = int(input())
```

```
FS = set(map(int,input().split()))

CS =ES.symmetric_difference(FS)
count = 0
for i in CS:
    count+=1
print(count)
```

set symmetric

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
if __name__ == "__main__":
    print(sum([pow(int(input()), int(input())), pow(int(input()), int(inp
ut()))]))
```

integers come in all sizes

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
if __name__ == "__main__":
    a, b = (int(input().strip()) for _ in range(2))
    print('{0}\n{1}\n{2}'.format(a // b, a % b, divmod(a, b)))
```

mod divmod

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
a,b,c = (int(input()), int(input()), int(input()))
print(pow(a,b))
print(pow(a,b,c))
```

power

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from cmath import *
import math

comm_num = complex(input())
x,y = comm_num.real,comm_num.imag
r, phi = math.sqrt(x ** 2 + y ** 2), phase(complex(x, y))
print(r, phi, sep="\n")
```

polar coordinates

```
a=list(map(int,input().split()))
b=list(map(int,input().split()))
res=[(x,y) for x in a for y in b]
print(*res)
```

itertools.product

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import itertools
a= list(map(str,input().split()))
p=sorted(list(itertools.permutations(a[0],int(a[1]))))
for i in p:
    print("".join(i))
```

itertools.permutations

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations
n, lst, k = int(input()),list(input().split(' ')), int(input())
cl = list(combinations(lst,k))
count = list(filter(lambda c: 'a' in c, cl))
print("{:.3f}".format(len(count)/len(cl)))
```

iterables and iterators

itertools with combinations and replacements

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations_with_replacement

text_str, size_int = input().split()
order_string = sorted(text_str)
iter_combination = combinations_with_replacement(order_string, int(size_i
nt))
[print(''.join(combination)) for combination in iter_combination]
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import Counter

n = int(input())
list1 = list(input().split())
dic1 = dict(Counter(list1))
cust = int(input())

total = 0
for i in range(cust):
    size, price = input().split()
    if size in dic1.keys():
        if dic1[size] != 0:
            dic1[size] -=1
            total+=int(price)
print(total)
```

collections counter

defaultdict tutorial

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import defaultdict

n, m = input().split()
words = defaultdict(list)
for i in range(int(n)):
    w = input().strip()
    words[w].append(i+1)

for _ in range(int(m)):
    bword = input().strip()
    print(*words[bword]) if words[bword] else print(-1)
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import deque
d = deque()
for _ in range(int(input())):
    cmd, *n = input().split()
    getattr(d, cmd)(*n)
print(*d)
```

```
collections.deque
```

piling up

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
for _ in range(int(input())):
    n = int(input())
    d = list(map(int,input().split()))
    flag = "Yes"
    for i in range(n // 2):
        if max(d[i],d[n-i-1]) < min(d[i + 1], d[n-i-2]):
            flag = "No"
    print(flag)
```