Viktoriia Vlasenko ADM1

Part 1

Introduction

Say "Hello, World!" With Python

```
print("Hello, World!")
```

Python If-Else

```
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    n = int(input().strip())
    #odd (1, 3, 5...) - weird
    if n%2!=0:
        print("Weird")
    #even (2, 4, 6...)
    else:
        if 2<=n<=5:
            print("Not Weird")
        elif 6<=n<=20:
            print("Weird")
        elif 20<=n:
            print("Not Weird")
```

Arithmetic Operators

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

Python: Division

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

Loops

```
if __name__ == '__main__':
    n = int(input())
    for i in range(n):
        print(i**2)
```

Write a function

```
def is_leap(year):
    leap = False
    if year % 4 == 0 and year % 100 != 0:
        leap = True
    elif year % 100 == 0 and year % 400 == 0:
        leap = True
    return leap
```

Print Function

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

Data types

List Comprehensions

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    result = []
    for i in range(x+1):
        for j in range(y+1):
            for k in range(z+1):
                if (i+j+k) != n:
```

```
                    result.append([i, j, k])
    print(result)
```

Find the Runner-Up Score!

```
if __name__ == '__main__':
    n = int(input())
    arr = [int(i) for i in input().split()]

    arr = sorted(arr)
    maxEl = arr[-1]

    runnerUp = arr[0]
    for i in arr:
        if runnerUp <= i and i != maxEl:
            runnerUp = i

    print(runnerUp)
```

Nested Lists

```
N = int(input())
records = []
for _ in range(N):
    st_name = input().strip()
    st_grade = float(input())
    records.append([st_name, st_grade])

grades = list(sorted([i[1] for i in records]))[::-1]
lowest = grades[-1]
while lowest in grades:
    grades.remove(lowest)
second_lowest = grades[-1]
names = []
for i in records:
    if i[1] == second_lowest:
        names.append(i[0])
names = sorted(names)
for name in names:
    print(name)
```

Finding the percentage

```
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()
```

```
    result = sum(student_marks[query_name])/len(student_marks[query_name])
    print("{:0.2f}".format(result))
```

Lists

```
if __name__ == '__main__':
    N = int(input())
    new_list = []
    commands = [input().strip().split() for i in range(N)]

    command_dict = {
        "insert": new_list.insert,
        "print": print,
        "remove": new_list.remove,
        "append": new_list.append,
    }

    for command in commands:
        params = [int(i) for i in command[1:]]
        if params:
            command_dict[command[0]](*params)
        elif command[0] == "print":
            command_dict[command[0]](new_list)
        elif command[0] == "sort":
            new_list.sort()
        elif command[0] == "pop":
            new_list.pop()
        elif command[0] == "reverse":
            new_list.reverse()
```

Tuples

```
if __name__ == '__main__':
    n = int(input())
    integer_list = map(int, input().split())
    print(hash(tuple(integer_list)))
```

Strings

sWAP cASE

```
def swap_case(s):
    swappedLine = ""
    for i in s:
        if i.islower():
            swappedLine += i.upper()
        else:
```

```
            swappedLine += i.lower()
    return swappedLine
```

String Split and Join

```
def split_and_join(line):
    line = "-".join(line.split(" "))
    return line

if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

What's Your Name?

```
def print_full_name(first, last):
    print("Hello {} {}! You just delved into python.".format(first, last))
```

Mutations

```
def mutate_string(string, position, character):
    return string[:position]+character+string[position+1:]
```

Find a string

```
def counter(index):
    if index != -1:
        return 1
    else:
        return 0

def count_substring(string, sub_string):
    num_sub = 0
    index = string.find(sub_string)
    num_sub += counter(index)
    while index != -1:
        index += 1
        index = string.find(sub_string, index+1, len(string))
        num_sub += counter(index)
    return num_sub
```

String Validators

```
if __name__ == '__main__':
    s = input()
    print (any(i.isalnum() for i in s))
    print (any(i.isalpha() for i in s))
    print (any(i.isdigit() for i in s))
    print (any(i.islower() for i in s))
    print (any(i.isupper() for i in s))
```

Text Alignment

```
#Replace all ______ with rjust, ljust or center.

thickness = int(input()) #This must be an odd number
c = 'H'

#Top Cone
for i in range(thickness):
    print((c*i).rjust(thickness-1)+c+(c*i).ljust(thickness-1))

#Top Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickness*6))

#Middle Belt
for i in range((thickness+1)//2):
    print((c*thickness*5).center(thickness*6))

#Bottom Pillars
for i in range(thickness+1):
    print((c*thickness).center(thickness*2)+(c*thickness).center(thickness*6))

#Bottom Cone
for i in range(thickness):
    print(((c*(thickness-i-1)).rjust(thickness)+c+(c*(thickness-i-1)).ljust(thickness)).rjust(thickness*6))
```

Text Wrap

```
def wrap(string, max_width):
    wrappedText = textwrap.wrap(string, width=max_width)
    return "\n".join(wrappedText)
```

Designer Door Mat

```
N, M = [int(i) for i in input().strip().split(" ")]
line_dots_pattern = ".|."
times = 1
center_line_num = N // 2
for i in range(N):
    if i < center_line_num:
        print((line_dots_pattern*times).center(M,'-'))
```

```
        times += 2
    elif i == center_line_num:
        print("WELCOME".center(M,'-'))
        times -= 2
    else:
        print((line_dots_pattern*times).center(M,'-'))
        times -= 2
```

String Formatting

```
def print_formatted(number):
    length = len(str(bin(number))) - 2
    for i in range(1, number+1):
        print(
            "{0:d}".format(i).rjust(length, ' '),
            "{0:o}".format(i).rjust(length, ' '),
            "{0:X}".format(i).rjust(length, ' '),
            "{0:b}".format(i).rjust(length, ' ')
        )
```

Alphabet Rangoli

```
def print_rangoli(size):
    temp = [chr(i) for i in range(ord('a'), ord('a')+size)]
    temp = temp[1:][::-1] + temp
    center = "-".join(temp)
    width = len(center)
    bottom = []
    for i in range(size - 1):
        bottom.append("-".join(temp[:len(temp)//2-i]+temp[len(temp)//2 + 2 + i:]).center(width, "-"))
    result = [*reversed(bottom), center, *bottom]
    for i in result:
        print(i)
```

Capitalize!

```
import textwrap

def solve(s):
    return " ".join([i.capitalize() for i in s.split(" ")])
```

The Minion Game

```
def minion_game(string):
    vowels = 'AEIOU'
    kevin = 0
    l = len(string)
```

```
    for i in range(l):
        if string[i] in vowels:
            kevin += l - i
    stuart = int(l * (l + 1) / 2) - kevin
    if stuart > kevin:
        print('Stuart', stuart)
        return
    elif kevin > stuart:
        print('Kevin', kevin)
        return
    else: print('Draw')
```

Merge the Tools!

```
def merge_the_tools(string, k):
    counter = 0
    subsets = []
    temp = []
    for i in string:
        counter += 1
        if i not in temp:
            temp.append(i)
        if counter == k:
            subsets.append("".join(temp))
            temp = []
            counter = 0
    for s in subsets:
        print(s)
```

Sets

Introduction to Sets

```
def average(array):
    array = set(array)
    return sum(array)/len(array)
```

No Idea!

```
n, m = [int(i) for i in input().strip().split(" ")]
arr = [int(i) for i in input().strip().split(" ")]
A = set([int(i) for i in input().strip().split(" ")])
B = set([int(i) for i in input().strip().split(" ")])

happiness = 0
for i in arr:
    if i in A:
        happiness += 1
    elif i in B:
```

```
        happiness -= 1

print(happiness)
```

Symmetric Difference

```
m = int(input())
m_set = {int(i) for i in input().split(" ")}
n = int(input())
n_set = {int(i) for i in input().split(" ")}

for i in sorted(m_set.symmetric_difference(n_set)):
    print(i)
```

Set .add()

```
n = int(input())
stamps = set([input() for i in range(n)])
print(len(stamps))
```

Set .discard(), .remove() & .pop()

```
n = int(input())
s = set(map(int, input().split()))
num_commands = int(input())

commands = {
    "remove": s.remove,
    "discard": s.discard,
    "pop": s.pop
}

for _ in range(num_commands):
    command = [i for i in input().strip().split(" ")]
    args = command[1:]
    if args:
        commands[command[0]](*[int(i) for i in args])
    else:
        commands[command[0]]()

print(sum(s))
```

Set .union() Operation

```
n = int(input())
n_set = set([int(i) for i in input().strip().split(" ")])
b = int(input())
```

```
b_set = set([int(i) for i in input().strip().split(" ")])

print(len(n_set.union(b_set)))
```

Set .intersection() Operation

```
n = int(input())
n_set = set([int(i) for i in input().strip().split(" ")])
b = int(input())
b_set = set([int(i) for i in input().strip().split(" ")])

print(len(n_set.intersection(b_set)))
```

Set .difference() Operation

```
n = int(input())
n_set = set([int(i) for i in input().strip().split(" ")])
b = int(input())
b_set = set([int(i) for i in input().strip().split(" ")])

print(len(n_set.difference(b_set)))
```

Set .symmetric_difference() Operation

```
n = int(input())
n_set = set([int(i) for i in input().strip().split(" ")])
b = int(input())
b_set = set([int(i) for i in input().strip().split(" ")])

print(len(n_set.symmetric_difference(b_set)))
```

Set Mutations

```
len_A = int(input())
A = set([int(i) for i in input().strip().split()])

N = int(input())
for i in range(N):
    current_command = [k for k in input().strip().split(" ")]
    otherSet = set([int(j) for j in input().strip().split(" ")])
    if current_command[0] == "intersection_update": A.intersection_update(otherSet)
    if current_command[0] == "update": A.update(otherSet)
    if current_command[0] == "symmetric_difference_update": A.symmetric_difference_update(otherSet)
    if current_command[0] == "difference_update": A.difference_update(otherSet)

print(sum(A))
```

The Captain's Room

```
K = int(input())
rooms = [int(i) for i in input().strip().split(" ")]
groups = {}
#print(len(rooms)//K)
for room in rooms:
    if room not in groups:
        groups[room] = 1
    else:
        groups[room] += 1
for k, v in groups.items():
    if v == 1:
        print(k)
```

Check Subset

```
T = int(input())
for i in range(T):
    len_A = int(input())
    A = set([int(j) for j in input().strip().split(" ")])
    len_B = int(input())
    B = set([int(j) for j in input().strip().split(" ")])
    print(A.issubset(B))
```

Check Strict Superset

```
A = set([int(i) for i in input().strip().split(" ")])
n = int(input())
otherSets = []
for i in range(n):
    otherSets.append([int(i) for i in input().strip().split(" ")])

cheks = []
for i in otherSets:
    if A.issuperset(set(i)):
        cheks.append(True)
    else:
        cheks.append(False)

print(sum(cheks)==len(cheks))
```

Collections

collections.Counter()

```
from collections import Counter

X = int(input())
stats = Counter(input().strip().split(" "))
N = int(input())
summ = 0
for _ in range(N):
    size, x = input().strip().split(" ")
    if stats[size] > 0:
        summ += int(x)
        stats[size] -= 1
print(summ)
```

DefaultDict Tutorial

```
n, m = [int(i) for i in input().strip().split(" ")]
letters = {
    "A": [],
    "B": []
}
for i in range(n):
    letters["A"].append(input().strip())

for i in range(m):
    letters["B"].append(input().strip())

for letter in letters["B"]:
    positions = []
    for pos in range(len(letters["A"])):
        if letter == letters["A"][pos]:
            positions.append(pos + 1)
    if positions:
        print(" ".join([str(i) for i in positions]))
    else:
        print(-1)
```

Collections.namedtuple()

```
from collections import namedtuple

N = int(input())
Student = namedtuple("Student", [i for i in input().split()])
summ = 0
for i in range(N):
    summ += int(Student(*[j for j in input().split()]).MARKS)
print(summ/N)
```

Collections.OrderedDict()

```
from collections import OrderedDict

N = int(input())
ordered_dictionary = OrderedDict()
for _ in range(N):
    item = input().strip().split(" ")
    name = " ".join(item[:-1])
    price = int(item[-1])
    if name in ordered_dictionary:
        ordered_dictionary[name] += price
    else:
        ordered_dictionary[name] = price

for key, value in ordered_dictionary.items():
    print(key, value)
```

Word Order

```
n = int(input())
d = {}
for _ in range(n):
    word = input().strip()
    if word in d:
        d[word] += 1
    else:
        d[word] = 1

print(len(set(d)))
for value in d.values():
    print(value, end=" ")
```

Collections.deque()

```
from collections import deque

d = deque()
commands = {
    "append": d.append,
    "appendleft": d.appendleft,
    "clear": d.clear,
    "extend": d.extend,
    "extendleft": d.extendleft,
    "count": d.count,
    "pop": d.pop,
    "popleft": d.popleft,
    "remove": d.remove,
    "reverse": d.reverse,
    "rotate": d.rotate
}

N = int(input())
for _ in range(N):
    command = input().strip().split(" ")
```

```
    arg = command[1:]
    if arg:
        commands[command[0]](*arg)
    else:
        commands[command[0]]()

print(*d)
```

Company Logo

```
from collections import Counter

s = sorted(input())
sets = Counter(s)
for key, value in sets.most_common(3):
    print(key, value)
```

Piling Up!

```
from collections import deque

T = int(input())
for i in range(T):
    n  = int(input())
    blocks = deque([int(a) for a in input().strip().split(" ")])
    result = []
    while blocks:
        if blocks[-1] >= blocks[0]:
            result.append(blocks.pop())
        else:
            result.append(blocks.popleft())
    if result == sorted(result, reverse=True):
        print('Yes')
    else:
        print('No')
```

Date and time

Calendar Module

```
import calendar

month, day, year = [int(i) for i in input().strip().split()]
print(str(calendar.day_name[calendar.weekday(year, month, day)]).upper())
```

Time Delta

```
import math
import os
import random
import re
import sys
from datetime import datetime

def time_delta(t1, t2):
    fmt = "%a %d %b %Y %H:%M:%S %z"
    t1_date = datetime.strptime(t1, fmt)
    t2_date = datetime.strptime(t2, fmt)
    if (t1_date.year + t2_date.year) <= 6000:
        return str(abs(int((t1_date - t2_date).total_seconds())))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    t = int(input())
    for t_itr in range(t):
        t1 = input()
        t2 = input()
        delta = time_delta(t1, t2)
        fptr.write(delta + '\n')
    fptr.close()
```

Exeptions

Exeptions

```
T = int(input())
for _ in range(T):
    try:
        a, b = [int(i) for i in input().strip().split(" ")]
        print(a//b)
    except ZeroDivisionError:
        print("Error Code: integer division or modulo by zero")
    except ValueError as e:
        print("Error Code:", e)
```

Built-ins

Zipped!

```
N, X = [int(i) for i in input().strip().split(" ")]
result = []
for _ in range(X):
    result.append([float(i) for i in input().strip().split(" ")])

for i in zip(*result):
    print(sum(i)/X)
```

Athlete Sort

```
import math
import os
import random
import re
import sys
from operator import itemgetter

if __name__ == '__main__':
    nm = input().split()
    n = int(nm[0])
    m = int(nm[1])
    arr = []
    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))
    k = int(input())
    attrs = sorted(arr, key= itemgetter(k))
    [print(*i) for i in attrs]
```

ginortS

```
lowercase = []
uppercase = []
odd = []
even = []

s = input()
for i in s:
    if i.islower(): lowercase.append(i)
    elif i.isupper(): uppercase.append(i)
    elif i.isdigit() and int(i) % 2 == 1: odd.append(i)
    elif i.isdigit() and int(i) % 2 == 0: even.append(i)

lowercase.sort()
uppercase.sort()
odd.sort()
even.sort()
print("".join(lowercase)+"".join(uppercase)+"".join(odd)+"".join(even))
```

Python Functionals

Map and Lambda Function

```
cube = lambda x: x**3

def fibonacci_el(n):
    if n in {0, 1}:
        return n
```

```
    return fibonacci_el(n - 1) + fibonacci_el(n - 2)

def fibonacci(n):
    result = []
    for i in range(n):
        result.append(fibonacci_el(i))
    return result
```

Regex and Parsing challenges

Detect Floating Point Number

```
import re

def if_float(num):
    pattern = r'^[+-]?\d*[.]\d+$'
    result = re.fullmatch(pattern, num)
    if result is not None:
        print(True)
    else:
        print(False)

N = int(input())
for _ in range(N):
    if_float(input())
```

Re.split()

```
regex_pattern = r"[,.]"
```

Group(), Groups() & Groupdict()

```
import re

string = input()
pattern = r'([A-Za-z0-9])\1+'
m = re.findall(pattern, string)
if m:
    print(m[0])
else:
    print(-1)
```

Re.findall() & Re.finditer()

```
import re

S = input()
pattern = r"(?<=[^aeiou])([aeiou]{2,})(?=[^aeiou])"
search = re.findall(pattern, S , flags=re.IGNORECASE)
if search:
    for s in search:
        print(s)
else:
    print(-1)
```

Re.start() & Re.end()

```
import re

string = input()
substring = input()
pattern = fr"{substring}"
index = 0
start = 0
end = 0
search = re.search(pattern, string)
if search == None: print("(-1, -1)")
while search:
    search = re.search(pattern, string)
    if search == None:
        break
    else:
        index = search.end() - 1
        string = string[index:]
        start = search.start() + end
        end = search.end() - 1 + end
        print(f"({start}, {end})")
```

Regex Substitution

```
import re

def sub_func(pattern, modified, line):
    while re.search(pattern, line):
        line = re.sub(pattern, modified, line)
    return line

N = int(input())
for _ in range(N):
    line = input()
    line = sub_func(r"(\s&&\s)", " and ", line)
    line = sub_func(r"(\s\|\|\s)", " or ", line)
    print(line)
```

Validating Roman Numerals

```
regex_pattern = r"^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$"
```

Validating phone numbers

```
import re

pattern = r'^[7|8|9]\d{9}'
N = int(input())
for _ in range(N):
    print("YES" if bool(re.fullmatch(pattern, input())) == True else "NO")
```

Validating and Parsing Email Addresses

```
import re
import email.utils

pattern = r'^[a-z][A-Za-z0-9\-\,\.\_]+@[A-Za-z]+\.[A-Za-z]{1,3}'
N = int(input())
for _ in range(N):
    s = input()
    parsed = email.utils.parseaddr(s)
    if bool(re.fullmatch(pattern, parsed[1])) == True:
        print(s)
```

Hex Color Code

```
import re

pattern = r'#(([a-fA-F0-9]){6}|([a-fA-F0-9]){3})'

N = int(input())
colors = []
for _ in range(N):
    line = input()
    search = re.findall(pattern, line)
    if len(line) != 0:
        if search and line[0] != "#":
            for i in search:
                colors.append(i[0])
for i in colors:
    print('#'+ i)
```

HTML Parser - Part 1

```
from html.parser import HTMLParser
```

```
class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Start :", tag)
        for key, value in attrs:
            print("->", key, ">", value)
    def handle_endtag(self, tag):
        print("End   :", tag)
    def handle_startendtag(self, tag, attrs):
        print("Empty :", tag)
        for key, value in attrs:
            print("->", key, ">", value)

parser = MyHTMLParser()
N = int(input())
for _ in range(N):
    parser.feed(input())
parser.close()
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_comment(self, data):
        if "\n" in data:
            print(">>> Multi-line Comment")
            data = data.split("\n")
            for d in data:
                print(d)
        else:
            print(">>> Single-line Comment")
            print(data)
    def handle_data(self, data):
        if data != "\n":
            print(">>> Data")
            print(data)

html = ""
N = int(input())
for i in range(N):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

Detect HTML Tags, Attributes and Attribute Values

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        for key, value in attrs:
```

```
            print("->", key, ">", value)
    def handle_startendtag(self, tag, attrs):
        print(tag)
        for key, value in attrs:
            print("->", key, ">", value)

parser = MyHTMLParser()
N = int(input())
for _ in range(N):
    parser.feed(input())
parser.close()
```

Validating UID

```
import re

def unique(s):
    d = {}
    for i in s:
        if i in d:
            d[i] += 1
        else:
            d[i] = 1
    check = True
    for k, v in d.items():
        if v > 1:
            check = False
    return check
```

Validating Credit Card Numbers

```
import re

def check_if_valid(number):
    temp_number = ""
    for i in number:
        if i == "-":
            continue
        else:
            temp_number += i
    number = temp_number
    pattern = r"(\d)\1{3}"
    search = re.search(pattern, number)
    if search == None:
        return "Valid"
    else:
        #print(number, ":", search.group())
        return "Invalid"

N = int(input())
pattern = r"(^[4|5|6][0-9]{3}\-?)([0-9]{4}\-?){3}$"
for _ in range(N):
    number = input()
    search = re.search(pattern, number)
    if search == None:
```

```
        print("Invalid")
    else:
        print(check_if_valid(number))
```

Validating Postal Codes

```
regex_integer_in_range = r"^[1-9]\d{5}$"  # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)" # Do not delete 'r'.
```

Matrix Script

```
import math
import os
import random
import re
import sys

first_multiple_input = input().rstrip().split()
n = int(first_multiple_input[0])
m = int(first_multiple_input[1])
matrix = []
for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)
a = list(zip(*matrix))
string = ""
for i in range(len(a)):
    string += "".join(a[i])
pattern = r"(?<=\w)[!@#$%& ]{1,}(?=\s*\w)"
search = re.sub(pattern, " ", string)
print(search)
```

XML

XML 1 - Find the Score

```
def get_attr_number(node):
    summury = 0
    for i in node.iter():
        summury += len(i.attrib)
    return summury
```

XML2 - Find the Maximum Depth

```
maxdepth = 0
def depth(elem, level):
    global maxdepth
    if (level == maxdepth):
        maxdepth += 1
    for child in elem:
        depth(child, level + 1)
```

Closures and Decorations

Standardize Mobile Number Using Decorators

```
def wrapper(f):
    def fun(l):
        f([f"+91 {i[-10:-5]} {i[-5:]}" for i in l])
    return fun
```

Decorators 2 - Name Directory

```
def person_lister(f):
    def inner(people):
        return map(f, sorted(people, key = lambda x: int(x[2])))
    return inner
```

Numpy

Arrays

```
def arrays(arr):
    return numpy.array(arr, float)[::-1]
```

Shape and Reshape

```
import numpy

n = m = 3
arr = [int(i) for i in input().strip().split(" ")]
print(numpy.reshape(arr, (n, m)))
```

Transpose and Flatten

```
import numpy

N, M = [int(i) for i in input().strip().split()]
arr = []
for i in range(N):
    arr.append([int(j) for j in input().strip().split()])
arr = numpy.array(arr)
print(arr.transpose())
print(arr.flatten())
```

Concatenate

```
import numpy

N, M, P = [int(i) for i in input().strip().split()]
array_1 = []
for i in range(N):
    array_1.append([int(j) for j in input().strip().split()])
array_2 = []
for i in range(M):
    array_2.append([int(j) for j in input().strip().split()])

print(numpy.concatenate((array_1, array_2)))
```

Zeros and Ones

```
import numpy

form = list(map(int, input().split()))
print(numpy.zeros(form, dtype = numpy.int))
print(numpy.ones(form, dtype = numpy.int))
```

Eye and Identity

```
import numpy
numpy.set_printoptions(legacy='1.13')
N, M = [int(i) for i in input().strip().split()]
print(numpy.eye(N, M))
```

Array Mathematics

```
import numpy

N, M = [int(i) for i in input().strip().split()]
```

```
a = numpy.array([input().split() for i in range(N)], dtype = int)
b = numpy.array([input().split() for i in range(N)], dtype = int)
print(a + b)
print(a - b)
print(a * b)
print(a // b)
print(a % b)
print(a ** b)
```

Floor, Ceil and Rint

```
import numpy
numpy.set_printoptions(legacy="1.13")

arr = numpy.array(input().strip().split(" "), float)
print(numpy.floor(arr))
print(numpy.ceil(arr))
print(numpy.rint(arr))
```

Sum and Prod

```
import numpy

N, M = [int(i) for i in input().strip().split(" ")]
arr = []
for i in range(N):
    arr.append([int(j) for j in input().strip().split(" ")])

arr = numpy.array(arr)
print(numpy.prod(numpy.sum(arr, axis = 0), axis=None))
```

Min and Max

```
import numpy

N, M = [int(i) for i in input().strip().split(" ")]
arr = []
for i in range(N):
    arr.append([int(j) for j in input().strip().split(" ")])
arr = numpy.array(arr)
print(numpy.max(numpy.min(arr, axis = 1)))
```

Mean, Var, and Std

```
import numpy

N, M = [int(i) for i in input().strip().split(" ")]
```

```
arr = []
for i in range(N):
    arr.append([int(j) for j in input().strip().split(" ")])
arr = numpy.array(arr)
print(numpy.mean(arr, axis = 1))
print(numpy.var(arr, axis = 0))
print(numpy.round(numpy.std(arr), 11))
```

Dot and Cross

```
import numpy

N = int(input())
A = numpy.array([[int(i) for i in input().strip().split(" ")] for j in range(N)])
B = numpy.array([[int(i) for i in input().strip().split(" ")] for j in range(N)])

print(numpy.dot(A, B))
```

Inner and Outer

```
import numpy

A = numpy.array([int(i) for i in input().strip().split(" ")])
B = numpy.array([int(i) for i in input().strip().split(" ")])

print(numpy.inner(A, B))
print(numpy.outer(A, B))
```

Polynomials

```
import numpy

P = [float(i) for i in input().strip().split(" ")]
x = float(input())

print(numpy.polyval(P, x))
```

Linear Algebra

```
import numpy

N = int(input())
A = [[float(j) for j in input().strip().split(" ")] for _ in range(N)]
print(numpy.round(numpy.linalg.det(A), 2))
```

Part 2

Birthday Cake Candles

```
import math
import os
import random
import re
import sys
from collections import Counter

def birthdayCakeCandles(candles):
    tallest = max(candles)
    return int(Counter(candles)[tallest])

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    candles_count = int(input().strip())
    candles = list(map(int, input().rstrip().split()))
    result = birthdayCakeCandles(candles)
    fptr.write(str(result) + '\n')
    fptr.close()
```

Number Line Jumps

```
import math
import os
import random
import re
import sys

def kangaroo(x1, v1, x2, v2):
    result = "NO"
    for jumps in range(10000):
        if x1 + jumps*v1 == x2 + jumps*v2:
            result = "YES"
    return result

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    first_multiple_input = input().rstrip().split()
    x1 = int(first_multiple_input[0])
    v1 = int(first_multiple_input[1])
    x2 = int(first_multiple_input[2])
    v2 = int(first_multiple_input[3])
    result = kangaroo(x1, v1, x2, v2)
    fptr.write(result + '\n')
    fptr.close()
```

Viral Advertising

```
import math
import os
import random
import re
import sys

def viralAdvertising(n):
    likes = 2
    cumulative = 2
    for i in range(n-1):
        likes = 3*likes//2
        cumulative += likes
    return cumulative

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    n = int(input().strip())
    result = viralAdvertising(n)
    fptr.write(str(result) + '\n')
    fptr.close()
```

Recursive Digit Sum

```
import math
import os
import random
import re
import sys

def superDigit(n, k):
    if len(n) == 1:
        return int(n)
    digits = list(n)
    super_digit = sum([int(i) for i in digits]) * k
    return superDigit(str(super_digit), 1)

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')
    first_multiple_input = input().rstrip().split()
    n = first_multiple_input[0]
    k = int(first_multiple_input[1])
    result = superDigit(n, k)
    fptr.write(str(result) + '\n')
    fptr.close()
```

Insertion Sort - Part 1

```
import math
import os
import random
import re
import sys
```

```
def insertionSort1(n, arr):
    for i in range(n - 1, 0, -1):
        if arr[i] < arr[i-1]:
            temp = arr[i]
            arr[i] = arr[i - 1]
            print(*arr)
            arr[i - 1] = temp
    print(*arr)

if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int, input().rstrip().split()))
    insertionSort1(n, arr)
```

Insertion Sort - Part 2

```
import math
import os
import random
import re
import sys

def insertionSort2(n, arr):
    for i in range(1, n):
        for j in range(i):
            if arr[j] > arr[i]:
                temp = arr[i]
                arr[i] = arr[j]
                arr[j] = temp
        print(*arr)

if __name__ == '__main__':
    n = int(input().strip())
    arr = list(map(int, input().rstrip().split()))
    insertionSort2(n, arr)
```