Homework1

September 28, 2022

1 INTRODUCTION

1) Say “Hello, World!” With Python

[ ]:
```
print("Hello, World!")
```

2) Python If-Else

[ ]:
```
n = int(input())

if n % 2 != 0:
    print("Weird")
elif n >= 2 and n <= 5:
    print("Not Weird")
elif n >= 6 and n <= 20:
    print("Weird")
else:
    print("Not Weird")
```

3) Arithmetic Operators

[ ]:
```
a = int(input())
b = int(input())

print(a+b)
print(a-b)
print(a*b)
```

4) Python: Division

[ ]:
```
a = int(input())
b = int(input())

print(a//b)
print(a/b)
```

5) Loops

```
[ ]: n = int(input())

     for i in range(0,n):
         print(i**2)
```

6) Write a function

```
[ ]: def is_leap(year):
         leap = False
         if year%4==0:
             leap=True
             if year%100==0:
                 leap=False
                 if year%400==0:
                     leap=True
         return leap
```

7) Print function

```
[ ]: n = int(input())

     print(*range(1,n+1), sep="")
```

2 DATA TYPES

1) List Comprehension

```
[ ]: x = int(input())
     y = int(input())
     z = int(input())
     n = int(input())

     lista = [[xx, yy, zz] for xx in range(x+1) for yy in range(y+1) for zz in␣
      ↪range(z+1) if xx + yy + zz != n]

     print(lista)
```

2) Find the Runner-Up Score!

```
[ ]: n=int(raw_input())

     lista=map(int, raw_input().split())

     print sorted(list(set(lista)))[-2]
```

3) Nested Lists

[ ]:
```
scores = []
records = []
names = []

for a in range(int(input())):
        name = input()
        score = float(input())
        scores.append(score)
        couples = [name, score]
        records.append(couples)
scores = list(set(scores))
scores.sort()
value = scores[1]
for couple in records:
        if couple[1] == value:
                names.append(couple[0])
names.sort()

for i in names:
        print(i)
```

4) Finding the percentage

[ ]:
```
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()
media = sum(student_marks[query_name]) / len(student_marks[query_name])

print("{:.2f}".format(media))
```

5) Lists

[ ]:
```
n = int(input())
list = []

for i in range(n):
    operazione = input()
    op_divisa = operazione.split(" ")
    if op_divisa[0] == "insert":
        posizione = int(op_divisa[1])
        valore = int(op_divisa[2])
        list.insert(posizione, valore)
    if op_divisa[0] == "print":
        print(list)
    if op_divisa[0] == "remove":
        valore = int(op_divisa[1])
        list.remove(valore)
    if op_divisa[0] == "append":
        valore = int(op_divisa[1])
        list.append(valore)
    if op_divisa[0] == "sort":
        list.sort()
    if op_divisa[0] == "pop":
        list.pop()
    if op_divisa[0] == "reverse":
        list.reverse()
```

6) Tuples

[ ]:
```
n = int(input())

integer_list = tuple(map(int, input().split()))

print(hash(integer_list))
```

3 STRINGS

1) sWAP cASE

[ ]:
```
def swap_case(s):
    swapped = s.swapcase()
    return swapped
```

2) String Split and Join

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

3) What's Your Name?

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

4) Mutations

[ ]:
```
def mutate_string(string, position, character):
    string2 = string[:position] + character + string[position+1:]
    return string2
```

5) Find a string

```
[ ]: def count_substring(string, sub_string):
         count = 0
         for i in range(len(string)):
             if string[i:].startswith(sub_string):
                 count += 1
         return count
```

6) String Validators

```
[ ]: s = input()

     is_num = False
     for char in s:
         if char.isalnum():
             is_num = True
             break
     print(is_num)

     is_alpha = False
     for char in s:
         if char.isalpha():
             is_alpha = True
             break
     print(is_alpha)

     is_digit = False
     for char in s:
         if char.isdigit():
             is_digit = True
             break
     print(is_digit)

     is_lower = False
     for char in s:
         if char.islower():
             is_lower = True
             break
     print(is_lower)

     is_upper = False
     for char in s:
         if char.isupper():
             is_upper = True
             break
     print(is_upper)
```

7) Text Alignment

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

thickness = int(input())
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))
```

8) Text Wrap

```
[ ]: def wrap(string, max_width):
    stringa = textwrap.fill(string, max_width)
    return stringa
```

9) Designer Door Mat

```
[ ]: size = list(map(int, input().split()))

for i in range(0, size[0]-1):
    if(i % 2 == 0):
        print(('.|.'*(i+1)).center(size[1], "-"))
print("WELCOME".center(size[1], "-"))
for i in range(0, size[0]-1):
    if(i % 2 == 0):
        print(('.|.'*(size[0]-2-i)).center(size[1], "-"))
```

10) String Formatting

```
[ ]: def print_formatted(number):
         space = len(bin(number)) - 2
         for i in range(number):
             dec = str(i + 1)
             octal = str(oct(i + 1))
             hexa = str(hex(i + 1))
             bina = str(bin(i + 1))
             print(dec.rjust(space), octal[2:].rjust(space), hexa[2:].upper().
      ↪rjust(space), bina[2:].rjust(space))
```

11) Alphabet Rangoli

```
[ ]: def print_rangoli(size):
         if size == 1:
             print("a")
         else:
             letters = "abcdefghijklmnopqrstuvwxyz"
             for i in range(1, size):
                 print(("-".join(letters[size-1:size-i-1:-1]) + "-" + "-".
      ↪join(letters[size-i+1:size])).center(size*2+(size-1)*2-1, "-"))
             print("-".join(letters[size-1:0:-1])+"-"+"-".join(letters[0:size:]))
             for i in range(1, size):
                 print(("-".join(letters[size-1:i-1:-1]) + "-" + "-".
      ↪join(letters[i+1:size])).center(size*2+(size-1)*2-1, "-"))
```

12) Capitalize!

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

13) The Minion Game

```
[ ]: def minion_game(string):
         vowels = "AEIOU"
         score_s = 0
         score_k = 0
         position = 0
         for i in string:
             if i in vowels:
                 score_k += len(string) - position
             else:
                 score_s += len(string) - position
             position += 1
         if score_k > score_s:
             print(f"Kevin {score_k}")
         elif score_s > score_k:
             print(f"Stuart {score_s}")
         else:
```

```
        print("Draw")
```

14) Merge the Tools!

```
[ ]: def merge_the_tools(string, k):
         letters = list(string)
         n = int(len(string)/k)
         splitted_list = [letters[x:x+k] for x in range(0, len(letters), k)]
         for i in range(n):
             final_list = list(dict.fromkeys(splitted_list[i]))
             print("".join(final_list))
```

4 SETS

1) Introduction to Sets

```
[ ]: def average(array):
         new_array = set(array)
         mean = sum(new_array)/len(new_array)
         return mean
```

2) No Idea!

```
[ ]: value = input().split()
     n = int(value[0])
     m = int(value[1])

     numbers = list(map(int, input().split()))

     A = set(map(int, input().split()))
     B = set(map(int, input().split()))

     happiness = 0
     for number in numbers:
         if number in A:
             happiness += 1
         if number in B:
             happiness -= 1
     print(happiness)
```

3) Symmteric Difference

```
[ ]: M = int(input())
     a = input()
     N = int(input())
     b = input()
```

```
a = set(a.split())
b = set(b.split())

output = list(map(int, a.symmetric_difference(b)))
output.sort()

for i in range(0, len(output)):
    print(output[i])
```

4) Set .add()

```
[ ]: n = int(input())
s = set()

for i in range(n):
    s.add(input())

print(len(s))
```

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

```
[ ]: n = int(input())
numbers = set(map(int, input().split()))
operations = int(input())

for i in range(operations):
    operation = list(input().split())
    if operation[0] == "pop":
        numbers.pop()
    if operation[0] == "remove":
        numbers.remove(int(operation[1]))
    if operation[0] == "discard":
        numbers.discard(int(operation[1]))

print(sum(numbers))
```

6) Set .union() Operation

```
[ ]: n = int(input())
group_A = set(input().split())
m = int(input())
group_B = set(input().split())

print(len(group_A.union(group_B)))
```

7) Set .intersection() Operation

```
[ ]: n = int(input())
     group_A = set(input().split())
     m = int(input())
     group_B = set(input().split())

     print(len(group_A.intersection(group_B)))
```

8) Set .difference() Operation

```
[ ]: n = int(input())
     group_A = set(input().split())
     m = int(input())
     group_B = set(input().split())

     print(len(group_A.difference(group_B)))
```

9) Set .symmetric_difference() Operation

```
[ ]: n = int(input())
     group_A = set(input().split())
     m = int(input())
     group_B = set(input().split())

     print(len(group_A.symmetric_difference(group_B)))
```

10) Set Mutations

```
[ ]: n = int(input())
     group_A = set(map(int, input().split()))
     n_sets = int(input())

     for i in range(n_sets):
         operation = list(input().split())
         group_B = set(map(int, input().split()))
         if operation[0] == "update":
             group_A.update(group_B)
         if operation[0] == "intersection_update":
             group_A.intersection_update(group_B)
         if operation[0] == "difference_update":
             group_A.difference_update(group_B)
         if operation[0] == "symmetric_difference_update":
             group_A.symmetric_difference_update(group_B)

     print(sum(group_A))
```

11) The Captain's Room

[ ]:
```
k = int(input())
num = list(map(int,input().split()))
num_set = set(num)
for i in num_set:
    num.remove(i)
num_set_2 = set(num)
print (list(num_set.difference(num_set_2))[0])
```

12) Check Subset

[ ]:
```
n = int(input())
for i in range(n):
    n_A = int(input())
    A = set(map(int, input().split()))
    n_B = int(input())
    B = set(map(int, input().split()))
    print(A.issubset(B))
```

13) Check Strict Superset

[ ]:
```
A = set(map(int, input().split()))
n = int(input())
test = True
for i in range(n):
    B = set(map(int, input().split()))
    if A.issuperset(B) == False:
        test = False
        break
print(test)
```

5 COLLECTIONS

1) collections.Counter()

[ ]:
```
x = int(input())
sizes = list(map(int, input().split()))
n = int(input())
earned = 0

for i in range(n):
    deal = list(map(int, input().split()))
    if deal[0] in sizes:
        earned += deal[1]
        sizes.remove((deal[0]))

print(earned)
```

2) DefaultDict Tutorial

[ ]:
```
from collections import defaultdict

n, m = list(map(int, input().split()))

dic_A = defaultdict(list)

for i in range(1, n + 1):
    dic_A[input()].append(i)

list_b = [input() for i in range(0, m)]

for b in list_b:
    if b in dic_A:
        print(*dic_A[b])
    else:
        print("-1")
```

3) Collections.namedtuple()

[ ]:
```
from collections import namedtuple

n, info = (int(input()), namedtuple("info", input().split()))
grades = [int(info._make(input().split()).MARKS) for _ in range(n)]
print(sum(grades)/len(grades))
```

4) Collections.OrderedDict()

[ ]:
```
from collections import OrderedDict

n = int(input())
item_dict = OrderedDict()
for i in range(n):
    item_name, net_price = input().rsplit(" ", 1)
    item_dict[item_name] = item_dict.get(item_name, 0) + int(net_price)
for i, j in item_dict.items():
    print(i, j)
```

5) Word Order

[ ]:
```
n = int(input())
words_dict = {}

for i in range(n):
    word = input()
    if word in words_dict:
        words_dict[word] += 1
    else:
        words_dict[word] = 1

print(len(words_dict))
for i in words_dict:
    print(words_dict[i], end= ' ')
```

6) Collections.deque()

```
[ ]: from collections import deque

n = int(input())
d = deque()

for i in range(n):
    command = list(input().split())
    if command[0] == "append":
        d.append(command[1])
    if command[0] == "pop":
        d.pop()
    if command[0] == "popleft":
        d.popleft()
    if command[0] == "appendleft":
        d.appendleft(command[1])

for i in range(len(d)):
    print(d[i], end = " ")
```

7) Company Logo

```
[ ]: from collections import Counter

letters = tuple(input())
counter = Counter(sorted(letters))
for i, j in counter.most_common(3):
        print(i, j)
```

8) Piling Up!

```
[ ]: T = int(input())

for _ in range(T):
    n = int(input())
    blocks = list(map(int, input().split()))
    blocks = list(dict.fromkeys(blocks))
    new_list = []
    for _ in blocks:
        validator = True
        if blocks[0] > blocks[-1]:
            new_list.append(blocks[0])
            blocks.remove(blocks[0])
        else:
            new_list.append(blocks[-1])
            blocks.remove(blocks[-1])
        if blocks[0] > blocks[-1]:
            new_added = blocks[0]
            blocks.remove(blocks[0])
        else:
            new_added = blocks[-1]
            blocks.pop()
        if new_added <= new_list[-1]:
            new_list.append(new_added)
        else:
            validator = False
            print("No")
            break
    if validator == True:
        print("Yes")
```

6 DATE AND TIME

1) Calendar Module

[ ]:
```
import calendar

month, day, year = map(int, input().split())
print(calendar.day_name[calendar.weekday(year, month, day)].upper())
```

2) Time Delta

[ ]:
```
import math
import os
import random
import re
import sys
from datetime import datetime

# Complete the time_delta function below.
def time_delta(t_1, t_2):
    t_1 = datetime.strptime(t_1, '%a %d %b %Y %H:%M:%S %z')
    t_2 = datetime.strptime(t_2, '%a %d %b %Y %H:%M:%S %z')
    return str(int(abs((t_1 - t_2).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()
```

7 EXCEPTIONS

1) Exceptions

[ ]:
```
t = int(input())

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

8 BUILT-INS

1) Zipped!

[ ]:
```
N, X = map(int, input().split())
total = []

for i in range(X):
    values = map(float, input().split())
    total.append(list(values))

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

2) Athlete Sort

[ ]:
```
import math
import os
import random
import re
import sys

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())

    arr.sort(key = lambda x : x[k])
    for i in arr:
        print(*i,sep=' ')
```

3) ginortS

[ ]:
```
s = list(input())
upper, lower, digit_even, digit_odd = [], [], [], []

for i in range(len(s)):
    if s[i].isupper() == True:
        upper.append(s[i])
    if s[i].islower() == True:
        lower.append(s[i])
    if s[i].isdigit() == True:
        if int(s[i]) % 2 == 0:
            digit_even.append(int(s[i]))
        else:
            digit_odd.append(int(s[i]))

for i in sorted(lower):
    print(i, end = "")
for i in sorted(upper):
    print(i, end = "")
for i in sorted(digit_odd):
    print(i, end = "")
for i in sorted(digit_even):
    print(i, end = "")
```

9 PYTHON FUNCTIONALS

1) Map and Lambda Function

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

def fibonacci(n):
    fib = [0, 1]
    if n == 0:
        return []
    elif n == 1:
        return [0]
    else:
        for i in range(1, n-1):
            fib.append(fib[i-1]+fib[i])
        return fib
```

10 REGEX AND PARSING CHALLENGES

1) Detect Floating Point Number

[ ]:
```
import re

T = int(input())

for i in range(T):
    string = input()
    try:
        float(string)
        pattern = "^[-|+]?\d*[.]\d+$"
        print(bool(re.match(pattern, string)))
    except:
        print("False")
```

2) Re.split()

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

3) Group(), Groups() & Groupdict()

[ ]:
```
import re

S = input()
finder = re.search(r"([a-z0-9A-Z])\1", S)
if finder == None:
    print("-1")
else:
    print(finder.groups()[0])
```

4) Re.findall() & Re.finditer()

[ ]:
```
import re

S = input()
pattern = "(?<=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])[aeiouAEIOU]{2,}(?
 ↪=[QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm])"
matches = re.findall(pattern, S)
```

```python
if len(matches) != 0:
    for match in matches:
        print(match)
else:
    print("-1")
```

5) Re.start() & Re.end()

[ ]:
```python
import re

S = input()
k = input()

m = re.search(k, S)

if m == None:
    print("(-1, -1)")
else:
    print(f"({m.start()}, {m.end() - 1})")
    finder = True
    if m.start() == m.end() - 1:
        start_pos = m.end()
    else:
        start_pos = m.end() - 1
    while finder == True:
        m = re.search(k, S[start_pos:])
        if m == None:
            finder = False
        else:
            print(f"({m.start() + start_pos}, {m.end() + start_pos - 1})")
            if m.start() == m.end() - 1:
                start_pos += m.end()
            else:
                start_pos += m.end() - 1
```

6) Regex Substitution

[ ]:
```python
import re

N = int(input())

for i in range(N):
    s = re.sub("(?<=\s)&&(?=\s)", "and", input())
    print(re.sub("(?<=\s)\|\|(?=\s)", "or", s))
```

7) 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})$"
```

8) Validating phone numbers

```
[ ]: import re

     N = int(input())

     for i in range(N):
            string = input()
            validator = re.search(r'^[789]\d{9}$', string)
            if validator != None:
              print('YES')
            else:
              print('NO')
```

9) Validating and Pairsing Email Addresses

```
[ ]: import email.utils
     import re

     n = int(input())
     pattern = r"^[A-Za-z].+[\@]{1}([A-Za-z])+[\.]{1}[a-z]{1,3}$" # I copied this␣
      ↪pattern

     for _ in range(n):
         string = input()
         validator = re.search(pattern, email.utils.parseaddr(string)[1])
         if validator != None:
             print(string)
```

10) Hex Color Code

```
[ ]: import re

     pattern = "(?<=[\s,:])(#([0-9A-Fa-f]{3}){1,2})(?!$)" # I copied this pattern
     lines = ""
     N = int(input())

     for _ in range(N):
         line = input()
         lines += (line)
     col_codes = re.findall(pattern, lines)

     for i in col_codes:
         print(i[0])
```

11) HTML Parser - Part 1

[ ]:

12) HTML Parser - Part 2

[ ]:

13) Detect HTML Tags, Attributes and Attribute Values

[ ]:

14) Validating UID

[ ]:

15) Validating Credit Card Numbers

[ ]:

16) Validating Postal Codes

[ ]:

17) Matrix Script

[ ]:

11 XML

1) XML 1 - Find the Score

[ ]:

2) XML2 - Find the Maximum Depth

[ ]:

12 CLOSURES AND DECORATIONS

1) Standardize Mobile Number Using Decorators

[ ]:
```
def wrapper(f):
    def fun(l):
        new_list = []
        for number in l:
            new_list.append('+91 ' + number[-10:-5] + ' ' + number[-5:])
        return f(new_list)
    return fun
```

2) Decorators 2 - Name Directory

```
[ ]: def person_lister(f):
         def inner(people):
             new_list = []
             for person in range(len(people)):
                 people[person][2] = int(people[person][2])
             people.sort(key = operator.itemgetter(2))
             for person in people:
                 new_list.append(f(person))
             return new_list
         return inner
```

13 NUMPY

1) Arrays

```
[ ]: def arrays(arr):
         new_arr = numpy.flip(numpy.array(arr, float))
         return new_arr
```

2) Shape and Reshape

```
[ ]: import numpy

     arr = numpy.array(input().strip().split(' '), int)
     new_arr = numpy.reshape(arr, (3, 3))
     print(new_arr)
```

3) Transpose and Flatten

```
[ ]: import numpy

     N, M = map(int, input().split())
     arr = numpy.array([input().split()], int)
     for i in range(1, N):
         new_row = numpy.array([input().split()], int)
         arr = numpy.vstack ((arr, new_row))

     print (numpy.transpose(arr))
     print (arr.flatten())
```

4) Concatenate

```
[ ]: import numpy

     N, M, P = map(int, input().split())
     arr_1 = numpy.array([input().split()], int)
```

```
for i in range(1, N):
    new_row_1 = numpy.array([input().split()], int)
    arr_1 = numpy.vstack ((arr_1, new_row_1))
arr_2 = numpy.array([input().split()], int)
for i in range(1, M):
    new_row_2 = numpy.array([input().split()], int)
    arr_2 = numpy.vstack ((arr_2, new_row_2))
print(numpy.concatenate((arr_1, arr_2)))
```

5) Zeros and Ones

[ ]:
```
import numpy

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

6) Eye and Identity

[ ]:
```
import numpy
numpy.set_printoptions(legacy = "1.13")

N, M = map(int, input().split())
print(numpy.eye(N, M))
```

7) Array Mathematics

[ ]:
```
import numpy

N, M = map(int, input().split())
arr_A = numpy.array([input().split()], int)
for i in range(1, N):
    new_row_A = numpy.array([input().split()], int)
    arr_A = numpy.vstack ((arr_A, new_row_A))
arr_B = numpy.array([input().split()], int)
for i in range(1, N):
    new_row_B = numpy.array([input().split()], int)
    arr_B = numpy.vstack ((arr_B, new_row_B))
print(numpy.add(arr_A, arr_B))
print(numpy.subtract(arr_A, arr_B))
print(numpy.multiply(arr_A, arr_B))
print(numpy.floor_divide(arr_A, arr_B))
print(numpy.mod(arr_A, arr_B))
print(numpy.power(arr_A, arr_B))
```

8) Floor, Ceil and Rint

[ ]:
```
import numpy
numpy.set_printoptions(legacy='1.13')

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

9) Sum and Prod

[ ]:
```
import numpy

N, M = map(int, input().split())
arr = numpy.array([input().split()], int)
for i in range(1, N):
    new_row = numpy.array([input().split()], int)
    arr = numpy.vstack ((arr, new_row))
print(numpy.prod(numpy.sum(arr, 0)))
```

10) Min and Max

[ ]:
```
import numpy

N, M = map(int, input().split())
arr = numpy.array([input().split()], int)
for i in range(1, N):
    new_row = numpy.array([input().split()], int)
    arr = numpy.vstack ((arr, new_row))
print(numpy.max(numpy.min(arr, 1)))
```

11) Mean, Var, and Std

[ ]:
```
import numpy

N, M = map(int, input().split())
arr = numpy.array([input().split()], int)
for i in range(1, N):
    new_row = numpy.array([input().split()], int)
    arr = numpy.vstack ((arr, new_row))
print(numpy.mean(arr, 1))
print(numpy.var(arr, 0))
print(numpy.around(numpy.std(arr), decimals=11))
```

12) Dot and Cross

[ ]:
```
import numpy

N = int(input())
```

```
arr_1 = numpy.array([input().split()], int)
for i in range(1, N):
    new_row_1 = numpy.array([input().split()], int)
    arr_1 = numpy.vstack ((arr_1, new_row_1))
arr_2 = numpy.array([input().split()], int)
for i in range(1, N):
    new_row_2 = numpy.array([input().split()], int)
    arr_2 = numpy.vstack ((arr_2, new_row_2))
print(numpy.dot(arr_1, arr_2))
```

13) Inner and Outer

[ ]:
```
import numpy

arr_1 = numpy.array(input().split(), int)
arr_2 = numpy.array(input().split(), int)
print(numpy.inner(arr_1, arr_2))
print(numpy.outer(arr_1, arr_2))
```

14) Polynomials

[ ]:
```
import numpy

P = numpy.array(input().split(), float)
x = int(input())
print(numpy.polyval(P, x))
```

15) Linear Algebra

[ ]:
```
import numpy

N = int(input())
arr = numpy.array([input().split()], float)
for i in range(1, N):
    new_row = numpy.array([input().split()], float)
    arr = numpy.vstack ((arr, new_row))
print(round(numpy.linalg.det(arr), 3))
```

14 ALGORITHMS CHALLENGES

1) Birthday Cake Candles

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'birthdayCakeCandles' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY candles as parameter.
#

def birthdayCakeCandles(candles):
    candle = max(candles)
    counter = 0
    for i in candles:
        if i == candle:
         counter += 1
    return counter

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()
```

2) Number Line Jumps

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'kangaroo' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. INTEGER x1
#  2. INTEGER v1
#  3. INTEGER x2
#  4. INTEGER v2
#

def kangaroo(x1, v1, x2, v2):
    if v2 - v1 == 0 or (x1 - x2) * (v2 - v1) < 0 or (x1 - x2) % (v2 - v1) != 0:
        return "NO"
    else:
        return "YES"

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()
```

3) Viral Advertising

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'viralAdvertising' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER n as parameter.
#
```

```
def viralAdvertising(n):
    liked = 0
    shared = 5
    for _ in range(n):
        new_likes = int(shared / 2)
        shared = new_likes * 3
        liked += new_likes
    return liked

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    result = viralAdvertising(n)

    fptr.write(str(result) + '\n')

    fptr.close()
```

4) Recursive Digit Sum

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'superDigit' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. STRING n
#  2. INTEGER k
#

def superDigit(n, k):
    tuple_n = tuple(str(sum(map(int, tuple(n))) * k))
    while len(tuple_n) > 1:
        tuple_n = tuple(str(sum(map(int, tuple_n))))
    return int(tuple_n[0])

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()
```

5) Insertion Sort - Part 1

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort1' function below.
#
# The function accepts following parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#

def insertionSort1(n, arr):
    number = arr[-1]
    arr_sorted = False
    while arr_sorted == False:
        if number < arr[n-2]:
            arr[n-1] = arr[n-2]
            [print(int(arr[x]), end = " ") for x in range(len(arr))]
            print()
            n -= 1
            if n == 1:
                arr[0] = number
                arr_sorted = True
        else:
            arr[n-1] = number
            arr_sorted = True
    [print(int(arr[x]), end = " ") for x in range(len(arr))]
```

```
if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort1(n, arr)
```

6) Insertion Sort - Part 2

[ ]:
```
#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort2' function below.
#
# The function accepts following parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#

def insertionSort2(n, arr):
    for i in range(len(arr)-1):
        if arr[i] > arr[i+1]:
            old = arr[i+1]
            arr[i+1] = arr[i]
            arr[i] = old
            if arr[i] != 0 and arr[i] < arr[i-1]:
                arr_sorted = False
            else:
                arr_sorted = True
            while arr_sorted == False:
                if i != 0:
                    if old < arr[i - 1]:
                        arr[i] = arr[i - 1]
                        arr[i - 1] = old
                        i -= 1
                    else:
                        arr[i] = old
                        arr_sorted = True
                else:
                    break
            [print(int(arr[x]), end=" ") for x in range(len(arr))]
        else:
            [print(int(arr[x]), end=" ") for x in range(len(arr))]
        print()

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort2(n, arr)
```