Part1

Say "Hello, World!" With Python

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

Python If-Else

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

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)
```

Loops

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

Write a function

```
def is_leap(year):
    return (year % 4 == 0
    and
     year % 100 != 0) or(year % 400 == 0)
```

Print Function

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

List Comprehensions

```
def check_sum(x, y, z, n):
    combinations = [[i, j, k] for i in range(x + 1) for j in range(y + 1) for k in range(z + 1) if i
+ j + k != n]
    print(combinations)

if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
check_sum(x, y, z, n)
```

Find the Runner-Up Score!

```
def runner_up(array_students):
  array_students = list(array_students)
  array_students.sort(reverse=True)
  for score in range(len(array_students)):
    if array_students[score] != array_students[score + 1]:
      runner_up = array_students[score + 1]
      break
    else:
        continue
  print(runner_up)

if __name__ == '__main__':
  n = int(input())
  arr = map(int, input().split())
  runner_up(arr)
```

Nested Lists

```
def sort_students(names_scores):
  scores = [student[1] for student in names_scores]
  sorted_scores = sorted(set(scores))

  if len(sorted_scores) < 2:
    print("Not enough unique scores.")
    return
  second_lowest = sorted_scores[1]

 names_second_lowest = [student[0] for student in names_scores if student[1] ==
second_lowest]

  for name in names_second_lowest:
    print(name)
if __name__ == '__main__':
  for _ in range(int(input())):
    name = input()
    score = float(input())
   names_scores.append((name, score))
```

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()
```

Lists

```
if __name__ == '__main__':
    N = int(input())
    arr = []
for i in range(N):
    s = input().split()
    for i in range(1, len(s)):
        s[i] = int(s[i])
    if s[0] == "append":
        arr.append(s[1])
    elif s[0] == "insert":
        arr.insert(s[1],s[2])
    elif s[0] == "remove":
        arr.remove(s[1])
    elif s[0] == "pop":
```

```
        arr.pop()
    elif s[0] == "sort":
        arr.sort()
    elif s[0] == "reverse":
        arr.reverse()
    elif s[0] == "print":
        print(arr)
```

Tuples

```
    n = int(input())
    integer_list = map(int, input().split())
    integer_list = tuple(integer_list)
    print(hash(integer_list))
```

sWAP cASE

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

String Split and Join

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

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 count_substring(string, sub_string):
    count = 0
    for i in range(0, len(string) - len(sub_string)+1):
        l = i
        for j in range(0, len(sub_string)):
            if string[l] == sub_string[j]:
                l += 1
                if j == len(sub_string)-1:
                    count = count + 1
                else:
                    continue
            else:
                    break
    return count
```

String Validators

```
if __name__ == '__main__':
    s = input()
    print(any([i.isalnum() for i in s]))
    print(any([i.isalnum() for i in s]))
    print(any([i.isalnum() for i in s]))
    print(any([i.isalnum() for i in s]))
    print(any([i.isalnum() for i in s]))
```

Text Alignment

```
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

```
import textwrap


def wrap(string, max_width):
   return textwrap.fill(string,max_width)
```

Designer Door Mat

```
n, m = int(input().split(' '))
for i in range(1,n,2):


print((".|."*i).center('_'))
print("welcom".center(m'_')


for i in range(n-2,-1,-2):
   print((".|."*i).center(m'_'))
```

String Formatting

```
def print_formatted(number):
   l = len(str(bin(number)[2:]))
   for i in range(1,number+1):
        print(str(i).rjust(l) + ' ' + oct(i)[2:].rjust(l) + ' ' + hex(i)[2:].upper().rjust(l) + ' ' + bin(i)[2:].rjust(l))
```

Alphabet Rangoli

```
def print_rangoli(size):
```

```
    # your code goes here
    alphabet = [chr(i) for i in range(97, 123)]
    alphabet = alphabet[:size]
    indices = list(range(size))
    indices = indices[:-1] + indices[::-1]
    for i in indices:
        start_index = i +1
        original = alphabet[-start_index:]
        reverse = original[::-1]
        row = reverse + original[1:]
        row = "-".join(row)
        width = size * 4 - 3
        row = row.center(width, "-")
        print(row)
```

Capitalize!

```
def solve(s):
    for n in s[:].split():
        s = s.replace(n, n.capitalize())
    return s
```

The Minion Game

```
def minion_game(string):
  vol = "AEIOU"
  Stuart_score, kevin_score = 0, 0
  length = len(string)
  for start_idx in range(length):
```

```
        score = length - start_idx
        if string[start_idx] in vol:
            kevin_score += score
        else:
            Stuart_score += score
    if Stuart_score == kevin_score:
        print('Draw')
    if Stuart_score > kevin_score:
        print('Stuart {}'.format(Stuart_score))
    if Stuart_score < kevin_score:
        print('Stuart {}'.format(kevin_score))
```

Merge the Tools!

```
def merge_the_tools(string, k):
  for part in zip(*[iter(string)] * k):
    d = dict()
    print(''.join([ d.setdefault(c, c) for c in part if c not in d]))
```

collections.Counter()

```
from collections import Counter
number_of_shoes = int(input())
shoe_size_list = list(map(int, input().split()))
number_of_customers = int(input())

counter_of_size = Counter(shoe_size_list)
total = 0
for customers in range(number_of_customers):
  size, price = list(map(int, input().split()))
  if size in counter_of_size.key():
```

```
    if counter_of_size[size] > 0:
      total = total + price
      counter_of_size[size] = counter_of_size[size] -1
print(total)
```

Introduction to Sets

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

DefaultDict Tutorial

```
from collections import defaultdict
d = defaultdict(list)
list1=[]
n, m = map(int, input().split())
for i in range(1, n+1):
  d[input()].append(str(i))
for i in range(m):
  b = input()
  if b in d: print(' '.join(d[b]))
  else: print(-1)
```

Calendar Module

```
import calendar
month, day, year = list(map(int, input().split()))

def day(month, day, year):
  weekday = calendar.weekday(year, month, day)
  weekday_cap = calendar.day_name[weekday].upper()
  return weekday_cap
```

```
result = day(month, day, year)
print(result)
```

Exceptions

```
items = int(input())
for i in range(items):
    try:
        a, b = input().split()
        result = int(a) // int(b)
        print(result)
    except ZeroDivisionError as e:
        print("Error Code:", e)
    except ValueError as v:
        print("Error Code:", v)
```

Collections.namedtuple()

```
from collections import namedtuple
n = int(input())
fields = input().split()
Student = namedtuple('Student', fields)
total_mark = 0
for _ in range(n):
    student_d = input().split()
    student = Student(*student_d)
    total_mark += int(student.total_mark)
average_mark = total_mark / n
```

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

Time Delta

```
    def time_delta(t1, t2):
        dete_format = "%a %d %b %Y %H:%M:%S %z"
        time1 = datetime.strptime(t1, date_format)
        time2 = datetime.strptime(t2, date_format)
        time_difference = abs(int((time1 - time2).total_second()))
        return str(time_difference)
```

No Idea!

```
n, m = input().split()
s_arr = input().split()
a = set(input().split())
b = set(input().split())
print(sum([(i in a) - (i in b) for i in s_arr]))
```

Collections.OrderedDict()

```
    from collections import OrderedDict
    d = OrderedDict()

    for _ in range(int(input())):
        item, space, quantity = input().rpartition(' ')
        d[item] = d.get(item, 0) + int(quantity)
    for item, quantity in d.items():
        print(item, quantity)
```

Symmetric Difference

```
 M = int(input())
n = set(map(int, input().split()))
a = int(input())
b = set(map(int, input().split()))
result = n.symmetric_difference(b)
sorted_symmetric_difference = sorted(symmetric_difference)
for item in symmetric_difference:
   print(item)
```

Set .add()

```
a = set()
[a.add(input()) for _ in range(int(input()))]
print(len(a))
```

Word Order

```
from collections import Counter, OrderedDict
class OrderedCounter(Counter, OrderedDict):
   pass
d = OrderedCounter(input() for _ in range(int(input())))
print(len(d))
print(*d.values())
```

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

```
n = int(input())
s = set(map(int, input().split()))
N = int(input())
lis = []
```

```
for i in range(N):
   x = input().split()
   lis.append(x)


for i in range(N):
   if lis[i][0] == 'pop':
     if lis(s) ==0:
        continue
       else:
         s.pop()
   if lis[i][0] == 'remove':
     x = lis[i][1]
     if int(x) in s:
        s.remove(int(x))
       else:
         continue
    if lis[i][0] == 'discard':
     x = lis[i][0]
     if int(x) in s:
        s.discard(int(x))
       else:
         continue
          s.discard()
print(sum(s))
```

Collections.deque()

```
from collections import deque
d = deque()
for _ in range(int(input())):
  inp = input().split()
  getattr(d, inp[0])(*[inp[1]] if len(inp) > 1 else [])
print(*[item for item in d])
```

Company Logo

```
from collections import Counter
import math
import os
import random
import re
import sys
if __name__ == '__main__':
  s = input()
  result = Counter(sorted(s)).most_common(3)
  for key, value in result:
    print(key, value)
```

Set .union() Operation

```
n = int(input())
l = list(input().split())
m = int(input())
k = list(input().split())
s1 = set(l)
s2 = set(k)
```

```
print(len(s1.union(s2)))
```

Piling Up!

```
for i in range(int(input())):
    input()
    lis = [int(i) for i in input().split()]
    min_lis = lis.index(min(lis))
    left = lis[:min_lis]
    right = lis[min_lis+1:]
    if left == sorted(left, reverse=True) and right == sorted(right):
        print("Yes")
    else:
        print("No")
```

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()
    if query_name in student_marks:
        x = ((float(student_marks[query_name][0]) + float(student_marks[query_name][1])
+ float(student_marks[query_name][2]))/3)
    print('%.2f' % x)
```

Set .intersection() Operation

```
num1, st1, num2, st2, = (set(input().split()) for i in range(4))
print(len(st1.intersection(st2)))
```

Set .difference() Operation

```
n1 = int(input())
set_1 = set(map(int, input().split()))
n2 = int(input())
set_2 = set(map(int, input().split()))
print(len(set_1-set_2))
```

Set .symmetric_difference() Operation

```
    _, a = input(), set(input().split())
    _, b =input(), set(input().split())
    print(len(a.symmetric_difference(b)))
```

Set Mutations

```
(_, A) = (int(input()), set(map(int, input().split())))
N = int(input())
for _ in range(N):
  (command, newset) = (input().split()[0], set(map(int, input().split())))
  getattr(A, command)(newset)
print(sum(A))
```

The Captain's Room

```
k, arr = int(input()), list(map(int, input().split()))
my_set = set(arr)
print(((sum(my_set)*k) - (sum(arr)))//(k-1))
```

```
    Check Subset
    T = int(input())
    for _ in range(T):
      input()
      a = set(input().split())
      input()
      b = set(input().split())
      print(a.issubset(b))

    Check Strict Superset
    a = set(input().split())
    print(all(a > set(input().split()) for _ in range(int(input()))))
    Zipped!
n, x = map(int, input().split())
marks = []

for i in range(x):
  course = list(map(float, input().split()))
  marks.append(course)

for students_marks in list(zip(*marks)):
  print(sum(students_marks)/x)
    Input()
    xk = list(map(int, input().split()))
    x = xk[0]
    k = xk[1]
```

```
p = input()
if eval(p) == k:
  print(True)
else:
  print(False)
```

Python Evaluation

```
x = input()
eval(x)
```

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())
  P = sorted(arr,key=lambda row:row[k])
  for i in range(len(P)):
    for j in range(len(P[i])):
      print(P[i][j], end=' ')
```

```
    print()
```

ginortS

```
lower = ""
upper = ""
odd = ""
even = ""
S = sorted(input())
for i in S:
  if i.islower():
    lower += i
  elif i.isupper():
    upper += i
  elif int(i) % 2 != 0:
    odd += i
  elif int(i) % 2 ==0:
    even += i
print(lower + upper + odd + even)
```

Detect Floating Point Number

```
import re
n = int(input())
p= r'^[+-]?[0-9]*\.[0-9]+$'
for i in range(n):
  s = input()
  print(bool(re.match(p, s)))
```

Map and Lambda Function

```
cube = lambda x: pow(x, 3)
# complete the lambda function


def fibonacci(n):
  n_list = [0,1]
  for i in range(2,n):
    n_list.append(n_list[i-2] + n_list[i-1])
  return(n_list[0:n])
```

Re.split()

```
regex_pattern = r"[.,]+"    # Do not delete 'r'.
import re
```

Hex Color Code

```
import re
N = int(input())

for i in range(0, N):
  S = input()
  X=S.split()
  if len(X)>1 and '{'not in X:
    X=re.findall(r'#[a-fA-F0-9]{3,6}',S)
    [print(i) for i in X]
```

HTML Parser - Part 1

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
  def handle_starttag(self, tag, attrs):
```

```
        print('Start :', tag)
        for attr in attrs:
            print('->',' > '.join(map(str, attr)))

    def handle_endtag(self, tag):
        print('End   :', tag)

    def handle_startendtag(self, tag, attrs):
        print('Empty :', tag)
        for attr in attrs:
            print('->',' > '.join(map(str, attr)))
html = ""
for i in range(int(input())):
    html += input()
parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

HTML Parser - Part 2

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
  def handle_comment(self, comment):
      if '\n' in comment:
          print('>>> Multi-line Comment')
      else:
          print('>>> Single-line Comment')
      print(comment)
```

```
 def handle_data(self, data):
    if data == '\n': return
    print('>>> Data')
    print(data)
```

Detect HTML Tags, Attributes and Attribute Values

```
from html.parser import HTMLParser
class MyHTMLParser(HTMLParser):
   def handle_starttag(self, tag, attrs):
      print(tag)
      [print('-> {} > {}'.format(*attr)) for attr in attrs]

html = '\n'.join([input() for _ in range(int(input()))])
parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

XML 1 - Find the Score

```
def get_attr_number(node):
   return(len(node.attrib) + sum(get_attr_number(child) for child in node))
   # your code goes here
```

Validating UID

```
import re
for _ in range(int(input())):
   U = ''.join(sorted(input()))
```

```
  try:
    assert re.search(r'[A-Z]{2}', U)
    assert re.search(r'\d\d\d', U)
    assert not re.search(r'[^a-zA-Z0-9]', U)
    assert not re.search(r'(.)\1', U)
    assert len(U) == 10
  except:
    print('Invalid')
  else:
    print('Valid')
```

Validating Credit Card Numbers

```
import re
Tester = re.compile(
  r"^"
  r"(?!.*(\d)(-?\1){3})"
  r"[456]"
  r"\d{3}"
  r"(?:-?\d{4}){3}"
  r"$")
for _ in range(int(input().strip())):
  print("Valid" if  Tester.search(input().strip()) else "Invalid")
```

XML2 - Find the Maximum Depth

```
maxdepth = -1
def depth(elem, level):
```

```
  global maxdepth
  # your code goes here
  if(level == maxdepth):
    maxdepth += 1

  for child in elem:
    depth(child, level + 1)
```

Standardize Mobile Number Using Decorators

```
def wrapper(f):
  def fun(l):
    # complete the function
    f(["+91 "+c[-10:-5]+" "+c[-5:] for c in l])
  return fun
```

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'.
```

Decorators 2 - Name Directory

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

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)
encoded_m = "".join([matrix[j][i] for i in range(m) for j in range(n)])
mat =r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])'
print(re.sub(mat,' ',encoded_m))
```

Arrays

```
import numpy
def arrays(arr):
   # complete this function
   # use numpy.array
 numpy_array = numpy.array(arr[::-1], float)
 return numpy_array
arr = input().strip().split(' ')
```

Shape and Reshape

```
import numpy as np
print(np.array(input().split(), int).reshape(3,3))
```

Transpose and Flatten

```
import numpy  as np
n, m = map(int, input().split())
l = []
for i in range(n):
   a = list(map(int, input().split()))
   l.append(a)
arr = np.array(l)
print(np.transpose(arr))
print(arr.flatten())
```

Concatenate

```
import numpy as np
a, b, c = map(int, input().split())
arr_1 = np.array([input().split() for _ in range(a)],int)
arr_2 = np.array([input().split() for _ in range(b)],int)
print(np.concatenate((arr_1,arr_2), axis=0))
```

Zeros and Ones

```
import numpy
nums = tuple(map(int, input().split()))
print(numpy.zeros(nums, dtype = numpy.int32))
print(numpy.ones(nums, dtype = numpy.int32))
```

Eye and Identity

```
import numpy as np
n, m = map(int, input().split())
```

```
    output = str(np.eye(n, m))
    print(output.replace("0", " 0").replace("1", " 1"))
```

Array Mathematics

```
import numpy as np
n, m = map(int, input().split())
a, b = (np.array([input().split() for _ in range(n)], dtype=int) for _ in range(2))
print(a+b, a-b, a*b, a//b, a%b, a**b, sep='\n')
```

Floor, Ceil and Rint

```
import numpy as np
l = list(map(float,input().split()))
arr=np.array(l)
np.set_printoptions(sign=" ")
print(np.floor(arr))
print(np.ceil(arr))
print(np.rint(arr))
```

Sum and Prod

```
import numpy
n, m = map(int, input().split())
a = numpy.array([input().split() for _ in range(n)], int)
print(numpy.prod(numpy.sum(a, axis=0)))
```

Min and Max

```
import numpy as np
n, m=map(int, input().split())
l = []
for i in range(n):
  list1=list(map(int, input().split()))
  l.append(list1)
```

```
arr=np.array(l)
print(np.max(np.min(arr, axis=1)))
```

Mean, Var, and Std

```
import numpy as np
n, m = [int(item) for item in input().split()]
arr = []
for i in range(n):
   arr.append([int(item) for item in input().split()])
print(np.mean(arr, axis=1))
print(np.var(arr, axis=0))
print(round(np.std(arr, axis=None),11))
```

Dot and Cross

```
import numpy
n = int(input())
a = numpy.array([input().split()for _ in range(n)], int)
b = numpy.array([input().split()for _ in range(n)], int)
ma = numpy.dot(a,b)
print(ma)
```

Inner and Outer

```
import numpy as np
a=np.array(input().split(),int)
b=np.array(input().split(),int)
print(np.inner(a,b))
print(np.outer(a,b))
```

Polynomials

```
import numpy as np
l=list(map(float, input().split()))
```

```
x=int(input())

arr=np.array(l)

print(np.polyval(arr,x))
```

Linear Algebra

```
import numpy as np

n = int(input())

a = np.array([input().split() for _ in range(n)],float)

np.set_printoptions(legacy='1.13')

print(np.linalg.det(a))
```

Part2:

Birth day candle cake

```
import os
def superDigits(n, k):
    if len(str(n)) == 1:
        return int(n)
    digit_sum = sum(int(digit) for digit in str(n))
    return superDigits(digit_sum * k, 1)
```

Number Line Jumps

```
import math
import os
import random
import re
import sys
def kangaroo(x1, v1, x2, v2):
    if v2 >= v1:
        return "NO"
    if (x1-x2)%(v2-v1)==0:
        return "YES"
    return "NO"
    # Write your code here
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):
    total_likes = 0
    shared = 5
    for i in range(n):
        like = shared // 2
        total_likes += like
        shared = like * 3

    return total_likes
    # Write your code here
```

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

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

    result = viralAdvertising(n)

    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):
    key = arr[-1]
    i = n-1
    while i > 0 and arr[i-1] > key:
        arr[i] = arr[i-1]
        print(*arr)
        i-= 1
    arr[i] = key
    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 j in range(1, n):
        key = arr[j]
        i = j
        while i > 0 and arr[i-1] > key:
            arr[i] = arr[i-1]
            i -= 1
        arr[i] = key
        print(*arr)
if __name__ == '__main__':
    n = int(input().strip())

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

    insertionSort2(n, arr)
```