**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ИНСТИТУТ ЦИФРЫ**

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Функции»

Студентки 2 курса, ФИТ-211 группы

**Колесник Полины Олеговны**

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Руководитель:

Доцент Зимин А. И.

Работа защищена

« »

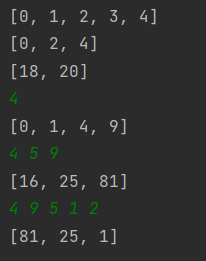
“ ” 2023 г.

Кемерово 2023 г.

**ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ**

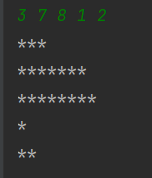
**1 задание**

my\_list = [i for i in range(5) if i < 5]  
print(my\_list)  
  
my\_list = [i for i in range(5) if i % 2 == 0]  
print(my\_list)  
  
my\_list = [int(i) \* 2 for i in range(11) if i \* 2 > 17]  
print(my\_list)  
  
my\_list = [int(i) \*\* 2 for i in range(int(input()))]  
print(my\_list)  
  
my\_list = [int(i) \*\* 2 for i in input().split()]  
print(my\_list)  
  
my\_list = [int(i) \*\* 2 for i in input().split() if int(i) % 2 != 0 and str(int(i) \*\* 2)[-1] != '9']  
print(my\_list)



**2 задание**

print("\n".join(int(number) \* "\*" for number in input().split()))



**3 задание**

def triangle(a, b, c):  
 if a + b > c and a + c > b and c + b > a:  
 print("Это треугольник")  
 else:  
 print("Это не треугольник")  
  
  
triangle(1, 1, 3)



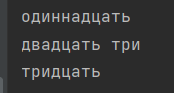
**4 задание**

import math  
  
  
def distance(x1, y1, x2, y2):  
 print(math.sqrt((y2 - y1) \*\* 2 + (x2 - x1) \*\* 2))  
  
  
distance(1, 3, 5, 6)



**5 задание**

def number\_to\_words(n):  
 slovar\_numbers = {  
 1: "один", 2: "два", 3: "три", 4: "четыре", 5: "пять",  
 6: "шесть", 7: "семь", 8: "восемь", 9: "девять",  
 11: "одиннадцать", 12: "двенадцать", 13: "тринадцать", 14: "четырнадцать", 15: "пятнадцать",  
 16: "шестнадцать", 17: "семнадцать", 18: "восемьнадцать", 19: "девятнадцать"  
 }  
  
 slovar\_dozens = {  
 10: "десять", 20: "двадцать", 30: "тридцать", 40: "сорок", 50: "пятьдесят",  
 60: "шестьдесят", 70: "семьдесят", 80: "восемьдесят", 90: "девяносто"  
 }  
  
 if n < 20:  
 print(slovar\_numbers[n])  
 elif n % 10 == 0:  
 print(slovar\_dozens[n])  
 elif n > 20 or n < 100:  
 print(slovar\_dozens[(n // 10) \* 10] + " " + slovar\_numbers[n % 10])  
  
  
number\_to\_words(11)  
number\_to\_words(23)  
number\_to\_words(30)



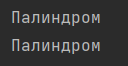
**6 задание**

def power(a, n):  
 degree = 1  
 for i in range(n):  
 degree \*= a  
 print(degree)  
  
  
power(2, 7)



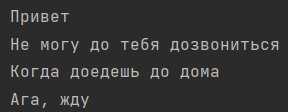
**7 задание**

def palindrome(s):  
 string = ''.join(s.split())  
  
 if len(string) % 2 == 0:  
 first\_part = string[:len(string) // 2]  
 second\_part = string[len(string) // 2:]  
 second\_part = second\_part[::-1]  
  
 if first\_part == second\_part:  
 print("Палиндром")  
 else:  
 print("Не палиндром")  
 else:  
 first\_part = string[:len(string) // 2].lower()  
 second\_part = string[len(string) // 2 + 1:].lower()  
 second\_part = second\_part[::-1]  
  
 if first\_part == second\_part:  
 print("Палиндром")  
 else:  
 print("Не палиндром")  
  
  
palindrome('1221')  
palindrome('А роза упала на лапу Азора')



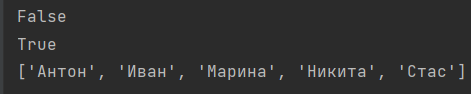
**8 задание**

text = ''  
  
  
def print\_without\_duplicates(message):  
 global text  
 if message != text:  
 print(message)  
 text = message  
  
  
print\_without\_duplicates('Привет')  
print\_without\_duplicates('Привет')  
print\_without\_duplicates('Не могу до тебя дозвониться')  
print\_without\_duplicates('Не могу до тебя дозвониться')  
print\_without\_duplicates('Не могу до тебя дозвониться')  
print\_without\_duplicates('Когда доедешь до дома')  
print\_without\_duplicates('Ага, жду')  
print\_without\_duplicates('Ага, жду')



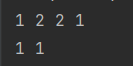
**9 задание**

person = {}  
  
  
def add\_friends(name\_of\_person, list\_of\_friends):  
 if person.get(name\_of\_person):  
 person[name\_of\_person] = person.get(name\_of\_person) + list\_of\_friends  
 else:  
 person[name\_of\_person] = list\_of\_friends  
  
  
def are\_friends(name\_of\_person1, name\_of\_person2):  
 if name\_of\_person2 in person[name\_of\_person1]:  
 return True  
 return False  
  
  
def print\_friends(name\_of\_person):  
 print(sorted(person[name\_of\_person]))  
  
  
add\_friends("Алла", ["Марина", "Иван"])  
print(are\_friends("Алла", "Никита"))  
add\_friends("Алла", ["Никита", "Антон", "Стас"])  
print(are\_friends("Алла", "Никита"))  
print\_friends("Алла")



**10 задание**

def mirror(arr):  
 mirrored\_part = reversed(arr)  
 arr += mirrored\_part  
  
  
arr = [1, 2]  
mirror(arr)  
print(\*arr)  
  
arr = [1]  
mirror(arr)  
print(\*arr)



**11 задание**

def from\_string\_to\_list(string, container):  
 container.extend(string.split())  
  
  
a = [1, 2, 3, 9, 40]  
from\_string\_to\_list("1 3 99 52", a)  
print(\*a)  
  
a = [77, 'abc']  
from\_string\_to\_list(" ", a)  
print(\*a)



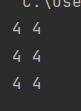
**12 задание**

def transpose(matrix):  
 n = len(matrix)  
 m = len(matrix[0])  
 return [[matrix[i][j] for i in range(n)] for j in range(m)]  
  
  
matrix = [[1, 2], [3, 4], [5, 6]]  
for line in transpose(matrix):  
 print(\*line)



**13 задание**

def matrix(\*args):  
 if len(args) == 0:  
 return [[0]]  
 elif len(args) == 1:  
 return [[0] \* args[0] for i in range(args[0])]  
 elif len(args) == 2:  
 return [[0] \* args[1] for i in range(args[0])]  
 elif len(args) == 3:  
 return [[args[2]] \* args[1] for i in range(args[0])]  
  
  
rows = matrix(3, 2, 4)  
for row in rows:  
 print(\*row)



**14 задание**

def partial\_sums(\*value):  
 res = [0]  
 for i in range(len(value)):  
 res.append(res[i] + value[i])  
 return res  
  
  
print(partial\_sums(13, 9, 4))



**15 задание**

def power(a, n):  
 if n == 1:  
 return a  
 return a \* power(a, n - 1)  
  
  
print(power(2, 5))



**16 задание**

def recurse\_len(some\_list):  
 if some\_list:  
 return 1 + recurse\_len(some\_list[1:])  
 else:  
 return 0  
  
  
print(recurse\_len([1, 2, 3]))



**17 задание**

def linear(some\_list):  
 if type(some\_list) is list:  
 if len(some\_list) == 1:  
 return linear(some\_list[0])  
 elif len(some\_list) > 1:  
 ans = []  
 ans1 = linear(some\_list[0])  
 ans.append(ans1) if type(ans1) is not list else ans.extend(ans1)  
 ans2 = linear(some\_list[1:])  
 ans.append(ans2) if type(ans2) is not list else ans.extend(ans2)  
 return ans  
 else:  
 return []  
 else:  
 return some\_list  
  
  
print(\*linear([[1, 2], 4, [[2, 4, 8, [-4, "er", [0], [{2: "primer"}], []]]]]))  
print(\*linear([[], [[], "EW"], [[[]]], [[[[[[[[[[[1, ]]], 25, ]]]]]]]]]))

