Министерство науки и высшего образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №1**

по дисциплине: «Исследование операций»

Тема: «Введение в язык программирования Python»

Выполнил:

студент группы

ПИН – 17.06

Завгороднев Е.Ю

Проверила:

ассистент кафедры ПО

Корнеева Е.И.

Тверь 2020

Оглавление

[Задача 1. Числа и записи выражений 3](#_Toc51582405)

[Задача 2. «Счастливый билет» 4](#_Toc51582406)

[Задача 3. Операции со списками 5](#_Toc51582407)

[Задача 4. Задача о черепашке 6](#_Toc51582408)

[Задача 5. Простой калькулятор 9](#_Toc51582409)

[Ссылка на репозиторий 11](#_Toc51582410)

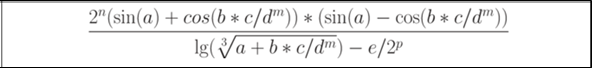
# Задача 1. Числа и записи выражений

Написать программу на языке Python 3.\*, которая вычисляет значение выражения (табл. 1 и табл. 2) и выводит результат на экран.

В таблице со сложными вариантами недостающие параметры вводятся с клавиатуры. Пример:

d = float(input("Введите параметр d\n"))

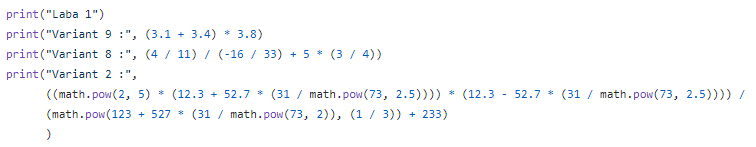
8.



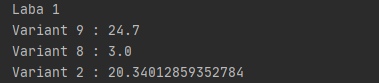
**Алгоритм:**

Разбиение выражения №9 на три компонента для более удобного занесения в программный код.

**Листинг программы:**



**Работа программы:**



**Вывод:**

В данной задаче использовал библиотеку math и научился работать с типом данных float.

# Задача 2. «Счастливый билет»

Билет общественного транспорта состоит из чётного числа цифр. При вводе нечётного числа – выводить фразу об ошибке.

Программа должна проверять равенство сумм первой и второй части номера билета. Если билет суммы равны, то билет «счастливый». Если суммы отличаются на 1, то билет «встречный».

На вход программе подаётся строка из четного числа цифр. Выводить нужно только слово «Счастливый», «Встречный» или «Обычный».

**Алгоритм:**

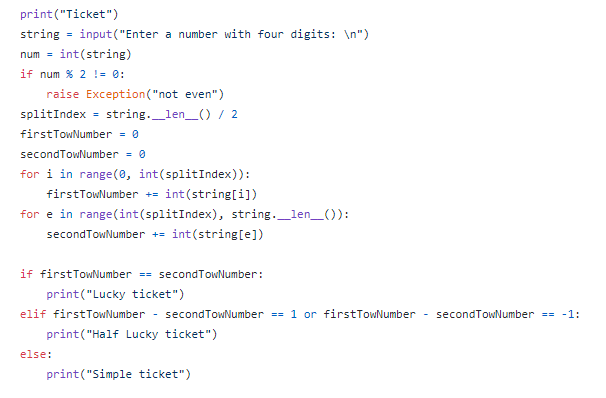
Ввод номера билета.

Проверка его на чётность.

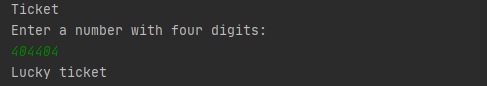
Разбиение номера на две части, с дальнейшим суммированием цифр в этих двух частях.

Проверка каким является билет.

**Листинг программы:**



**Работа программы:**



**Вывод:**

В данной задаче рассмотрел свойства списков в python и ветвление.

# Задача 3. Операции со списками

А. Задать с клавиатуры размер массива целых чисел.

Б. Ввести в цикле данные в массив.

В. Вывести минимальное, максимальное числа и сумма цифр в массиве.

Г. Сгенерировать массив 2 и повторить для него пункт В.

Д. Создать массив с помощью библиотеки numpy и методов numpy.zeros, numpy.arange, numpy.array.

**Алгоритм:**

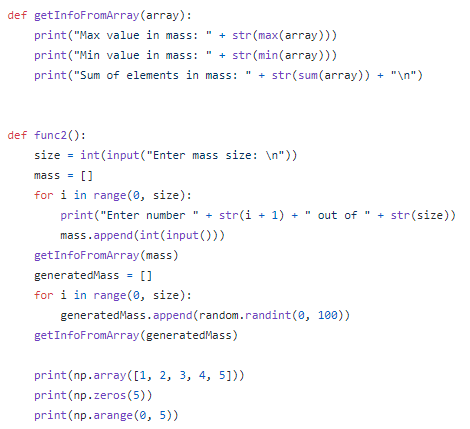
Ввод размера массива.

Заполнение его вручную.

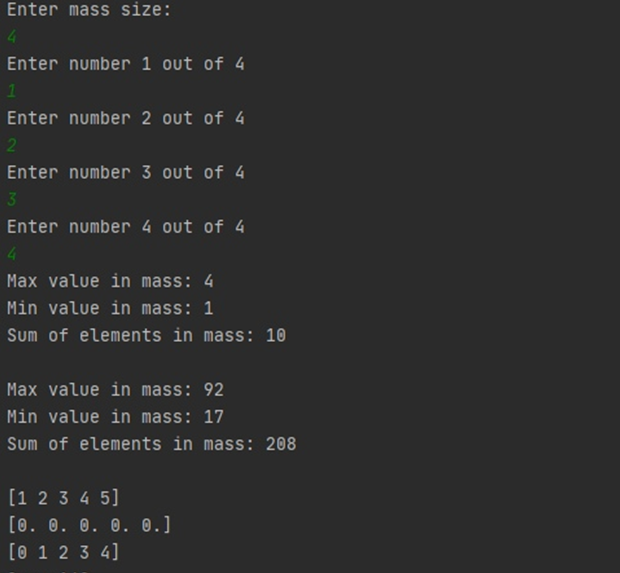
Вывод мин, макс элемента и суммы элементов.

Создание массива с помощью библиотеки numpy.

**Листинг программы:**



**Работа программы:**



**Вывод:**

В данной задаче рассмотрел методы библиотеки numpy. В отличии стандартных «массивов», нельзя изменять размер и данные должны быть одно и того же типа. Zeros возвращает массив, указанного размера заполненный нулями. Arrange массив с числами заданного промежутка по возрастанию. Array создаётся массив. Для более детального исследования лучше всего подходят массивы numpy.

## 

# Задача 4. Задача о черепашке

Дано прямоугольное поле . Клетки поля заполнены целыми числами. Черепашка находится в левой верхней клетке (координаты матрицы 0, 0). За один ход она перемещается в соседнюю клетку либо вправо, либо вниз (влево и вверх перемещаться запрещено). Стоимость каждой из клеток прибавляется к очкам Черепашки.

Найдите максимальную стоимость пути Черепашки (максимальное число баллов).

**Алгоритм:**

Чтение массива из файла со стоимостями. Проход по массиву с суммированием пути.

**Листинг программы:**

1. **import** numpy as np
2. **import** random
3. **import** os

6. **def** readFile(fileName):
7. filehandle **=** open(fileName)
8. file **=** filehandle.read()
9. mass **=** file.split("\n")
10. twoDimMass **=** []
11. **for** i **in** range(0, len(mass)):
12. temp **=** mass[i].split(" ")
13. temp1 **=** []
14. **for** j **in** range(0, len(temp)):
15. temp1.append(int(temp[j]))
16. twoDimMass.append(temp1)
17. filehandle.close()
18. **return** twoDimMass

21. **def** task3():
22. twoDimensionalArray **=** readFile('input1.txt')
23. sum **=** twoDimensionalArray[0][0]
24. step **=** Step(0, 0)
25. move **=** Move
26. array **=** []
27. **while** len(twoDimensionalArray) !**=** step.x **+** 1 **or** len(twoDimensionalArray[0]) !**=** step.y **+** 1:
28. step **=** move.doStep(twoDimensionalArray, step)
29. array.append(step)
30. **for** item **in** array:
31. sum **+=** twoDimensionalArray[item.x][item.y]
32. print("Sum: " **+** str(sum))

35. **class** Move:
36. **def** doStep(twoDimensionalArray, step):
37. sumDown **=** goDown(twoDimensionalArray, step)
38. sumRight **=** goRight(twoDimensionalArray, step)
39. **if** sumDown > sumRight:
40. **return** Step(step.x **+** 1, step.y)
41. **elif** sumDown < sumRight:
42. **return** Step(step.x, step.y **+** 1)
43. **elif** sumRight **==** sumRight **and** sumRight > 0:
44. **return** Step(step.x, step.y **+** 1)
45. **else**:
46. **return** Step(step.x, step.y)

49. **def** goDown(twoDimensionalArray, step):
50. **if** len(twoDimensionalArray) > step.x **+** 1:
51. **return** twoDimensionalArray[step.x **+** 1][step.y]
52. **else**:
53. **return** 0

56. **def** goRight(twoDimensionalArray, step):
57. **if** len(twoDimensionalArray[step.x]) > step.y **+** 1:
58. **return** twoDimensionalArray[step.x][step.y **+** 1]
59. **else**:
60. **return** 0

63. **class** Step:
65. **def** \_\_init\_\_(self, x, y):
66. self.x **=** x
67. self.y **=** y
69. **def** getXposition(self):
70. **return** self.x
72. **def** getYposition(self):
73. **return** self.y
75. **def** setXposition(self, x):
76. self.x **=** x
78. **def** setYposition(self, y):
79. self.y **=** y

82. **if** \_\_name\_\_ **==** '\_\_main\_\_':
83. task3()

**Работа программы:**



**Вывод:**

В данной задаче рассмотрел работу классов и модуля для работы с операционной системой в python.

# Задача 5. Простой калькулятор

Напишите простой калькулятор, который считывает с пользовательского ввода три строки: первое число, второе число и операцию, после чего применяет операцию к введённым числам ("первое число" "операция" "второе число") и выводит результат на экран.

Поддерживаемые операции: +, -, /, \*, mod, pow, div.   
mod – это взятие остатка от деления,   
pow – возведение в степень,   
div — целочисленное деление.

Если выполняется деление и второе число равно 0, необходимо выводить строку "Деление на 0!".

**Алгоритм:**

Инициализация окно графического приложения (свойства, функции).

Ввод переменных и операции над ними.

Вычисление результата.

Сохранение результата в файл.

**Листинг программы:**

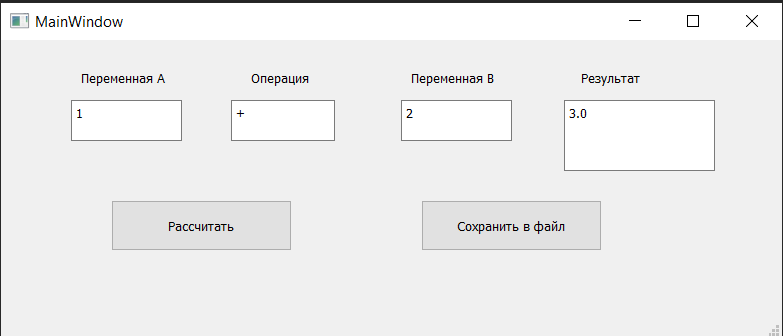
1. **from** PyQt5 **import** uic, QtCore, QtGui, QtWidgets, Qt
2. **import** sys

5. Form, \_ **=** uic.loadUiType("Calculator.ui")

8. **class** Ui(QtWidgets.QMainWindow, Form):
9. **def** \_\_init\_\_(self):
10. super(Ui, self).\_\_init\_\_()
11. self.setupUi(self)
12. self.Calc.clicked.connect(self.Calcul)
13. self.SaveF.clicked.connect(self.SvFile)
15. **def** SvFile(self):
16. f **=** open("calculator's answer.txt", "w")
17. f.write(str(self.Result.toPlainText()))
18. f.close()
20. **def** Calcul(self):
21. **try**:
22. res **=** 0.0
23. num1 **=** float(self.InputA.toPlainText())
24. num2 **=** float(self.InputB.toPlainText())
25. op **=** str(self.Operation.toPlainText())
26. **if** op **==** "+":
27. res **=** num1 **+** num2
28. self.Result.setPlainText(str(res))
29. **elif** op **==** "-":
30. res **=** num1 **-** num2
31. self.Result.setPlainText(str(res))
32. **elif** op **==** "\*":
33. res **=** num1 **\*** num2
34. self.Result.setPlainText(str(res))
35. **elif** op **==** "pow" **or** op **==** "\*\*":
36. res **=** num1 **\*\*** num2
37. self.Result.setPlainText(str(res))
38. **elif** op **==** "/":
39. res **=** num1 **/** num2
40. self.Result.setPlainText(str(res))
41. **elif** op **==** "mod" **or** op **==** "%":
42. res **=** num1 **%** num2
43. self.Result.setPlainText(str(res))
44. **elif** op **==** "div" **or** op **==** "//":
45. res **=** num1 **//** num2
46. self.Result.setPlainText(str(res))
47. **else**:
48. self.Result.setPlainText(str("Операция не поддерживается в калькуляторе"))
49. **except** ZeroDivisionError:
50. self.Result.setPlainText(str("Деление на ноль!"))
51. **except**:
52. self.Result.setPlainText(str("Ошибка !"))

55. **if** \_\_name\_\_ **==** "\_\_main\_\_":
56. app **=** QtWidgets.QApplication(sys.argv)
57. w **=** Ui()
58. w.show()
59. sys.exit(app.exec\_())

**Работа программы:**



**Вывод:**

В данной задаче научился использовать возможности PyQT Designer.

Ссылка на репозиторий: https://github.com/Dead-Inside-Coders/Operations-research