**Отчёт по лабораторной работе№1.**

**Программирование. Язык Python. Введение.**

**Лабораторная работа № 1. Задачи.**

**Комплект 1: Установка среды программирования и разработки.**

1.1: Установите Python последней версии и менеджер пакетов PIP одним из предложенных в Интернете способов (или убедитесь, что PIP уже установлен).

1.2: Заведите аккаунт на Replit.com для лабораторных проектов/заданий на Python.

**Комплект 2: Задачи для самостоятельного изучения материала.**

2.1: Изучите документы по оформлению программного кода PEP8. Изучите документы по оформлению документации к коду

2.2: Изучите руководство Replit.com для начинающих:.

**Комплект 3: Задачи для самостоятельной работы.**

3.1: Создайте простую программу калькулятор, которая позволяет из функции main() ввести два числа и тип арифметической операции, а потом вычисляет результат. Свой код опубликуйте на Replit.com и предоставьте ссылку в ответах на лабораторную работу в Moodle в документе-отчёте. Реализацию арифметических действий и вычисление результата с его возвратом сделайте в отдельной функции calculate(...). Протестируйте свой калькулятор с помощью вызова нескольких своих простых функций test\_∗() сключевым словом assert внутри. Обязательно напишите хорошую документацию к своему коду.

**Листинг программы:**

print ("Начнем калькулировать")

x1 = int (input("Введите число 1 "))

x2 = int (input("Введите число 2 "))

v = int (input("Какую операцию вы хотите выполнить? \n 1 Сложение \n 2 Вычитание \n 3 Деление \n 4 Умножение \n"))

if v == 1:

r = x1 + x2

p = "сложения"

t = p

if v == 2:

r = x1 - x2

l = "вычитания"

t = l

if v == 3:

if x2 != 0:

r = float(x1 / x2)

m = "деления"

t = m

else:

print("Делить на 0 нельзя")

if v == 4:

r = x1 \* x2

n = "умножения"

t = n

print ("Результат" ,t ,r)

def test\_addition():

x1 = 5

x2 = 3

v = 1

expected\_result = 8

r = x1 + x2

assert r == expected\_result, f"Ошибка в сложении: {x1} + {x2} должно быть {expected\_result}, а получено {r}"

def test\_subtraction():

x1 = 10

x2 = 4

v = 2

expected\_result = 6

r = x1 - x2

assert r == expected\_result, f"Ошибка в вычитании: {x1} - {x2} должно быть {expected\_result}, а получено {r}"

def test\_division():

x1 = 15

x2 = 3

v = 3

expected\_result = 5.0

r = x1 / x2

assert r == expected\_result, f"Ошибка в делении: {x1} / {x2} должно быть {expected\_result}, а получено {r}"

def test\_multiplication():

x1 = 4

x2 = 6

v = 4

expected\_result = 24

r = x1 \* x2

assert r == expected\_result, f"Ошибка в умножении: {x1} \* {x2} должно быть {expected\_result}, а получено {r}"

def test\_division\_by\_zero():

x1 = 10

x2 = 0

v = 3

try:

r = x1 / x2

except ZeroDivisionError:

pass

else:

assert False, "Ошибка: деление на 0 должно вызывать исключение ZeroDivisionError"

# Запуск всех тестов

test\_addition()

test\_subtraction()

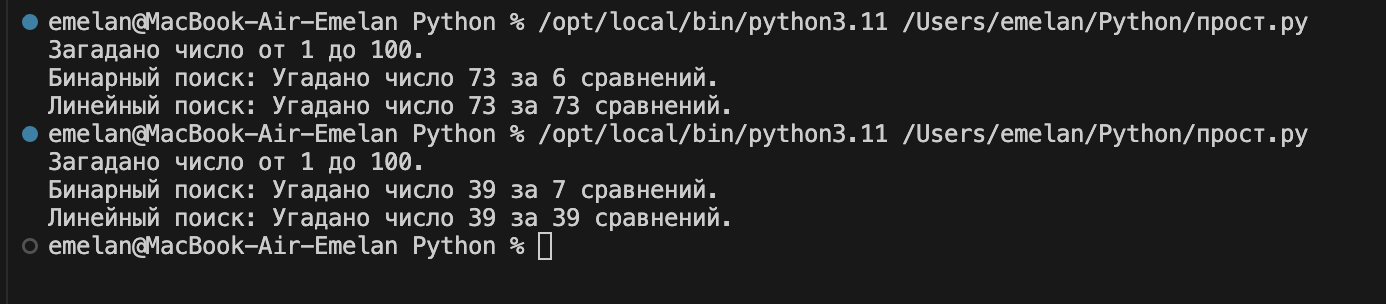
test\_division()

test\_multiplication()

test\_division\_by\_zero()

print("Все тесты пройдены успешно!")

**Результаты работы программы**



3.2: Реализуйте программно классическую простую игру "угадай число" (guess number) с помощью алгоритма медленного перебора (инкремента) по одному числа, либо с помощью алгоритма бинарного поиска. Алгоритм принимает на вход само число, которое он должен угадать, интервал значений в котором оно загадано и в цикле делает угадывания тем или иным выбранным вами способом. После угадывания из функции алгоритма возвращается угаданное число и число угадываний/сравнений, которые пришлось проделать. Обязательно напишите хорошую документацию к своему коду.

**Листинг программы**

import random

def guess\_number\_binary\_search(target\_number, min\_value=1, max\_value=100):

"""

Угадывает загаданное число с помощью бинарного поиска.

Args:

target\_number (int): Число, которое нужно угадать.

min\_value (int, optional): Минимальное значение диапазона. По умолчанию 1.

max\_value (int, optional): Максимальное значение диапазона. По умолчанию 100.

Returns:

int, int: Угаданное число и количество сравнений.

"""

num\_comparisons = 0

while min\_value <= max\_value:

guess = (min\_value + max\_value) // 2

num\_comparisons += 1

if guess == target\_number:

return guess, num\_comparisons

elif guess < target\_number:

min\_value = guess + 1

else:

max\_value = guess - 1

return -1, num\_comparisons

def guess\_number\_linear\_search(target\_number, min\_value=1, max\_value=100):

"""

Угадывает загаданное число с помощью линейного перебора.

Args:

target\_number (int): Число, которое нужно угадать.

min\_value (int, optional): Минимальное значение диапазона. По умолчанию 1.

max\_value (int, optional): Максимальное значение диапазона. По умолчанию 100.

Returns:

int, int: Угаданное число и количество сравнений.

"""

num\_comparisons = 0

for i in range(min\_value, max\_value + 1):

num\_comparisons += 1

if i == target\_number:

return i, num\_comparisons

return -1, num\_comparisons

if \_\_name\_\_ == "\_\_main\_\_":

# Пример использования

target\_number = random.randint(1, 100)

print(f"Загадано число от {1} до {100}.")

# Бинарный поиск

result, comparisons = guess\_number\_binary\_search(target\_number)

if result == -1:

print("Ошибка: не удалось угадать число.")

else:

print(f"Бинарный поиск: Угадано число {result} за {comparisons} сравнений.")

# Линейный поиск

result, comparisons = guess\_number\_linear\_search(target\_number)

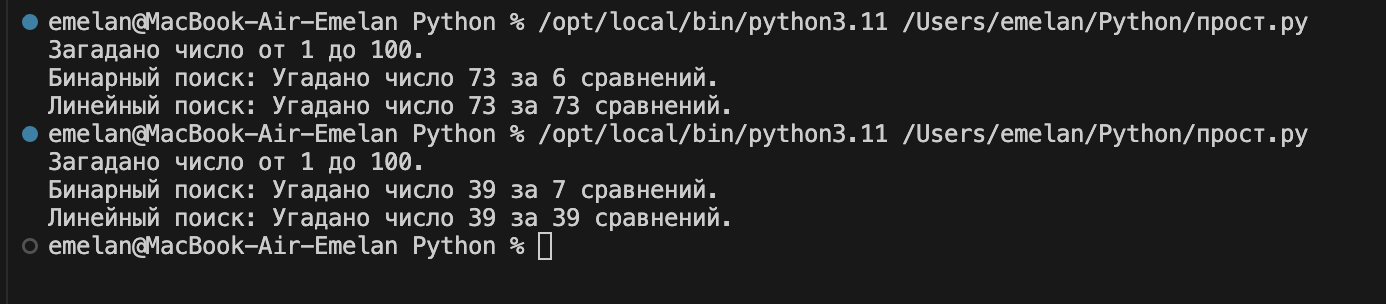
if result == -1:

print("Ошибка: не удалось угадать число.")

else:

print(f"Линейный поиск: Угадано число {result} за {comparisons} сравнений.")

**Результат работы программы**

****

Примечание.

В ответе должно быть количество попыток, а не количество угадываний.