Возврат значений из функции return, алгоритмы с заданной точностью для расчёта

Цели практической работы

- Научиться:
  - возвращать результат из функции — return;
  - передавать значения, объекты в функции через аргументы;
  - использовать несколько операторов return в функции;
  - использовать алгоритмы с заданной точностью для расчёта.
- Изучить способы возврата значений из функции — return.
- Узнать:
  - о передаче объектов в функцию и их поведении при изменении внутри функции;
  - об особенностях сравнения чисел с плавающей точкой.

Задача 1. Урок информатики 2

В прошлый раз учитель написал программу, которая выводит числа в формате плавающей точки, однако он вспомнил, что не учёл одну важную вещь: числа-то могут идти от нуля.

Задано положительное число x (x > 0). Ваша задача — преобразовать его в формат плавающей точки, то есть x = a * 10^b, где 1 ≤ a < 10. Обратите внимание, что x теперь больше нуля, а не больше единицы. Обеспечьте контроль ввода.

Пример 1:

Введите число: 92345

Формат плавающей точки: x = 9.2345 * 10 ** 4

Пример 2:

Введите число: 0.0012

Формат плавающей точки: x = 1.2 * 10 ** -3

```
def floating_point(number):
  count = 0
  if number >= 10:
    while number >= 10:
      number /= 10
      count += 1
    number = round((number), count)

  elif 0 < number < 1:
    while number < 1:
      number *= 10
      count += 1
    number = round((number), count)
  print('Формат плавающей точки: x =', number, '* 10 **', count)
```

```
number = float(input('Введите число: '))
if number <= 0:
  print('Ошибка ввода. Число должно быть больше нуля.')
else:
  floating_point(number)
```

Задача 2. Функция максимума

Юра пишет различные полезные функции для Python, чтобы остальным программистам стало проще работать. Он захотел написать функцию, которая будет находить максимум из перечисленных чисел. Функция для нахождения максимума из двух чисел у него уже есть. Юра задумался: может быть, её можно как-то использовать для нахождения максимума уже от трёх чисел?

Помогите Юре написать программу, которая находит максимум из трёх чисел. Для этого используйте только функцию нахождения максимума из двух чисел.

По итогу в программе должны быть реализованы две функции:

1. maximum_of_two — функция принимает два числа и возвращает одно (наибольшее из двух);
2. maximum_of_three — функция принимает три числа и возвращает одно (наибольшее из трёх); при этом она должна использовать для сравнений первую функцию maximum_of_two.

```
def maximum_of_two(number_1, number_2):
  if number_1 > number_2:
    return number_1
  else:
    return number_2

def maximum_of_three(max_number, number_3):
  if max_number > number_3:
    return max_number
  else:
    return number_3

number = [int(input('Введите число: ')) for i in range(3)]
number_1 = number[0]
number_2 = number[1]
number_3 = number[2]

max_number = maximum_of_two(number_1, number_2)
print('Максимальное число:', maximum_of_three(max_number, number_3))
```

```
def maximum_of_two(number_1, number_2):
  if number_1 > number_2:
    return number_1
  else:
    return number_2

def maximum_of_three(max_number, number_3):
  if max_number > number_3:
    return max_number
  else:
    return number_3
```

```
number_1 = int(input('Введите первое число: '))
number_2 = int(input('Введите второе число: '))
number_3 = int(input('Введите третье число: '))

max_number = maximum_of_two(number_1, number_2)
print('Максимальное число:', maximum_of_three(max_number, number_3))
```

Задача 3. Число наоборот 2

Пользователь вводит два числа: N и K. Напишите программу, которая заменяет каждое число на число, которое получается из исходного записью его цифр в обратном порядке, затем складывает их, снова переворачивает и выводит ответ на экран.

Пример:

Введите первое число: 102
Введите второе число: 123

Первое число наоборот: 201
Второе число наоборот: 321
Сумма: 522

Сумма наоборот: 225

```
def turn_over(num):
  count = 0
  while num > 0:
    count = count * 10 + num % 10
    num = num // 10
  return count

n = int(input('Введите первое число: '))
k = int(input('Введите второе число: '))

summ = turn_over(n) + turn_over(k)

print('Первое число наоборот:', turn_over(n))
print('Второе число наоборот:', turn_over(k))
print('Сумма:', summ)
print('Сумма наоборот:', turn_over(summ))
```

Задача 4. Недоделка 2

Вы всё так же работаете в конторе по разработке игр и смотрите различные программы прошлого горе-программиста. В одной из игр для детей, связанной с мультяшной работой с числами, вам нужно было написать код согласно следующим условиям: программа получает на вход два числа; в первом числе должно быть не менее трёх цифр, во втором — не менее четырёх, иначе программа выдаёт ошибку. Если всё нормально, то в каждом числе первая и последняя цифры меняются местами, а затем выводится их сумма.

И тут вы натыкаетесь на программу, которая была написана предыдущим программистом и которая как раз решает такую задачу. Однако старший программист попросил вас немного переписать этот код, чтобы он не выглядел так ужасно. Да и вам самим становится, мягко говоря, не по себе от него.

Постарайтесь разделить логику кода на три отдельные логические части (функции):

1. count_numbers — получает число и возвращает количество цифр в числе;
2. change_number — получает число, меняет в нём местами первую и последнюю цифры и возвращает изменённое число;
3. main — функция ничего не получает на вход, внутри она запрашивает нужные данные от пользователя, выполняет дополнительные проверки и вызывает функции 1 и 2 для выполнения задачи (проверки и изменения двух чисел).

Разбейте приведённую ниже программу на функции. Повторений кода должно быть как можно меньше. Также сделайте, чтобы в основной части программы был только ввод чисел, затем изменённые числа и вывод их суммы.

```
first_n = int(input("Введите первое число: "))

first_num_count = 0
temp = first_n

while temp > 0:
    first_num_count += 1
    temp = temp // 10
if first_num_count < 3:
 print("В первом числе меньше трёх цифр.")
else:
 last_digit = first_n % 10
 first_digit = first_n // 10 ** (first_num_count - 1)
 between_digits = first_n % 10 ** (first_num_count - 1) // 10
 first_n = last_digit * 10 ** (first_num_count - 1) + between_digits * 10 +
first_digit
 print('Изменённое первое число:', first_n)

 second_n = int(input("\nВведите второе число: "))

 second_num_count = 0
 temp = second_n
 while temp > 0:
   second_num_count += 1
   temp = temp // 10

 if second_num_count < 4:
   print("Во втором числе меньше четырёх цифр.")
 else:
   last_digit = second_n % 10
   first_digit = second_n // 10 ** (second_num_count - 1)
   between_digits = second_n % 10 ** (second_num_count - 1) // 10
   second_n = last_digit * 10 ** (second_num_count - 1) + between_digits * 10
+ first_digit

   print('Изменённое второе число:', second_n)

   print('\nСумма чисел:', first_n + second_n)
print('Задача 4. Недоделка 2\n')

def count_numbers(first_n, second_n):
```

```
    temp = first_n
    first_num_count = 0
    temp2 = second_n
    second_num_count = 0

    while temp > 0:
        first_num_count += 1
        temp = temp // 10

    while temp2 > 0:
        second_num_count += 1
        temp2 = temp2 // 10
    return first_num_count, second_num_count


def change_number(first_n, first_num_count, second_n, second_num_count):
    last_digit = first_n % 10
    first_digit = first_n // 10**(first_num_count - 1)
    between_digits = first_n % 10**(first_num_count - 1) // 10
    first_n = last_digit * 10**(first_num_count - 1) + between_digits * 10
+ first_digit
    print('Изменённое первое число:', first_n)
    last_digit = second_n % 10
    first_digit = second_n // 10 ** (second_num_count - 1)
    between_digits = second_n % 10 ** (second_num_count - 1) // 10
    second_n = last_digit * 10 ** (second_num_count - 1) + between_digits
* 10 + first_digit
    print('Изменённое второе число:', second_n)
    return first_n, second_n


def main():
    first_n = int(input("Введите первое число: "))
    second_n = int(input("\nВведите второе число: "))
    first_num_count, second_num_count = count_numbers(first_n , second_n)
    if first_num_count < 3:
        print("В первом числе меньше трёх цифр.")
        print(first_n)
    if second_num_count < 4:
        print("Во втором числе меньше четырёх цифр.")
        print(second_n)
    else:
       first_n, second_n = change_number(first_n, first_num_count,
second_n, second_num_count)
       print('\nСумма чисел:',first_n + second_n)

main()
```

Задача 5. Маятник

Известно, что амплитуда качающегося маятника с каждым разом затухает на 8,4% от амплитуды предыдущего колебания. Если качнуть маятник, то, строго говоря, он

не остановится никогда, просто амплитуда будет постоянно уменьшаться до тех пор, пока мы не сочтём такой маятник остановившимся. Напишите программу, определяющую, сколько раз качнётся маятник, прежде чем он, по нашему мнению, остановится.

Программа получает на вход начальную амплитуду колебания в сантиметрах и конечную амплитуду колебаний, которая считается остановкой маятника. Обеспечьте контроль ввода.

Пример:

Введите начальную амплитуду: 1

Введите амплитуду остановки: 0.1

Маятник считается остановившимся через 27 колебаний

```
# Ввод начальной амплитуды
initial_amplitude = float(input("Введите начальную амплитуду: "))
# Ввод амплитуды остановки
stop_amplitude = float(input("Введите амплитуду остановки: "))

# Проверка корректности ввода
if initial_amplitude <= 0:
    print("Начальная амплитуда должна быть положительным числом.")
    exit()
if stop_amplitude <= 0:
    print("Амплитуда остановки должна быть положительным числом.")
    exit()
if stop_amplitude >= initial_amplitude:
    print("Амплитуда остановки должна быть меньше начальной амплитуды.")
    exit()

# Коэффициент затухания (8,4% уменьшения, значит оставшаяся часть 91.6%)
decay_factor = 0.916

# Инициализация переменных
amplitude = initial_amplitude
count = 0

# Цикл, пока амплитуда не станет меньше или равна конечной
while amplitude > stop_amplitude:
    amplitude *= decay_factor
    count += 1

print(f"Маятник считается остановившимся через {count} колебаний")

Без exit
print('Задача 5. Маятник')

initial_amplitude = float(input("Введите начальную амплитуду: "))
stop_amplitude = float(input("Введите амплитуду остановки: "))

if initial_amplitude <= 0:
```

```
  print("Начальная амплитуда должна быть положительным числом.")
elif stop_amplitude <= 0:
  print("Амплитуда остановки должна быть положительным числом.")
elif stop_amplitude >= initial_amplitude:
  print("Амплитуда остановки должна быть меньше начальной амплитуды.")
else:
  decay_factor = 0.916

amplitude = initial_amplitude
count = 0

while amplitude > stop_amplitude:
  amplitude *= decay_factor
  count += 1

print(f"Маятник считается остановившимся через {count} колебаний")
```

Задача 6. Яйца

В рамках программы колонизации Марса компания «Спейс Инжиниринг» вывела особую породу черепах, которые, по задумке, должны размножаться, откладывая яйца в марсианском грунте. Откладывать яйца слишком близко к поверхности опасно из-за радиации, а слишком глубоко — из-за давления грунта и недостатка кислорода. Вообще, факторов очень много, но специалисты проделали большую работу и предположили, что уровень опасности для черепашьих яиц рассчитывается по формуле: D = x^3 − 3x^2 − 12x + 10, где x — глубина кладки в метрах, а D — уровень опасности в условных единицах. Для тестирования гипотезы нужно взять пробу грунта на безопасной, согласно формуле, глубине.

Напишите программу, находящую такое значение глубины x, при котором уровень опасности как можно более близок к нулю. На вход программе подаётся максимально допустимое отклонение уровня опасности от нуля, а программа должна рассчитать приблизительное значение x, удовлетворяющее этому отклонению. Известно, что глубина точно больше нуля и меньше четырёх метров. Обеспечьте контроль ввода.

Пример:

Введите максимально допустимый уровень опасности: 0.01

Приблизительная глубина безопасной кладки: 0.732421875 м

Что оценивается

- Результат вывода соответствует условию.
- Input содержит корректное приглашение для ввода.
- Формат вывода соответствует примеру.
- Вывод содержит описание результата (выведенные числа сопровождаются текстовым описанием).

```
# Запрашиваем у пользователя максимально допустимое отклонение
tolerance = float(input("Введите максимально допустимое отклонение уровня опасности от нуля: "))

# Инициализируем переменные для хранения лучшего результата
best_x = 0.0  # начнем с 0
best_diff = 10**9  # очень большое число, чтобы сравнивать

# Перебираем значения x от 0 до 4 с шагом 0.001
x = 0.0
found = False  # флаг, что нашли подходящее значение
while x <= 4:
    # Вычисляем уровень опасности для текущего x
    D = x**3 - 3*x**2 - 12*x + 10
    # Находим абсолютное значение разницы с нулём
    diff = abs(D)
    # Проверяем, если разница меньше или равна допустимому отклонению и лучше предыдущих результатов
    if diff <= tolerance and diff < best_diff:
        best_diff = diff
        best_x = x
        found = True
    x += 0.001

# После цикла проверяем, нашли ли подходящее значение
if found:
    print(f"Приблизительная глубина безопасной кладки: {best_x:.9f} м")
else:
    print("Не удалось найти подходящую глубину в заданных пределах и с заданной точностью.")

# Функция для вычисления уровня опасности (можно определить прямо перед основным кодом)
def level_of_danger(x):
    return x**3 - 3*x**2 - 12*x + 10
def level_of_danger(x):
    return x**3 - 3*x**2 - 12*x + 10

tolerance = float(input("Введите максимально допустимый уровень опасности (от нуля): "))

best_x = 0
best_diff = 10**9
x = 0.0
found = False

while x < 4:
  D = x**3 - 3*x**2 - 12*x + 10
  diff = abs(D)
  if diff <= tolerance and diff < best_diff:
    best_diff = diff
    best_x = x
    found = True
```

```
  # print(x)
  x += 0.0001

if found:
  print(f"Приблизительная глубина безопасной кладки: {best_x:.9f} м")
else:
  print("Ошибка!")
```