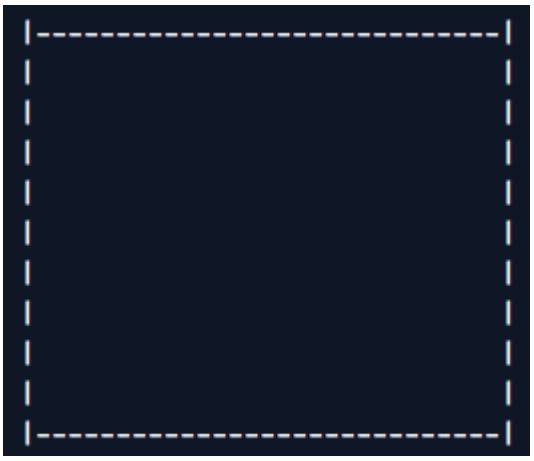
Погружение в Python (семинары)

Задание 1. Рамка

Напишите программу, которая рисует прямоугольную рамку с помощью символьной графики. Для вертикальных линий используйте символ вертикального штриха (|), а для горизонтальных — дефис (-). Пусть ширину и высоту рамки определяет пользователь.



Подсказка № 1

Используйте цикл `for` для создания строк рамки. Каждая итерация этого цикла будет соответствовать одной строке.

Подсказка № 2

Внутри внешнего цикла используйте ещё один цикл `for` для создания символов в строке. Каждая итерация этого цикла будет соответствовать одному символу в строке.

Подсказка № 3

Используйте условные выражения `if` для определения, какой символ выводить: вертикальную линию (`|`), горизонтальную линию (`-`) или пробел ( ).

Эталонное решение:

```
# Пользователь вводит размеры рамки

width = int(input("Введите ширину рамки: "))

height = int(input("Введите высоту рамки: "))
```

```
# Внешний цикл отвечает за строки (высота)
for line in range(height + 1):
    # Внутренний цикл отвечает за столбцы (ширина)
    for col in range(width + 1):
        # Если текущий столбец - первый или последний, рисуем вертикальную границу
        if col == width or col == 0:
            print('|', end='')
        # Если текущая строка - первая или последняя, рисуем горизонтальную границу
        elif line == height or line == 0:
            print('-', end='')
        # В остальных случаях оставляем пробелы
        else:
            print(' ', end='')
    # Завершаем текущую строку и переходим к следующей
    print()
```

Задача 2. Треугольник

Треугольник существует только тогда, когда сумма любых двух его сторон больше третьей. Дано a, b, c - стороны предполагаемого треугольника. Требуется сравнить длину каждого отрезка-стороны с суммой двух других. Если хотя бы в одном случае отрезок окажется больше суммы двух других, то треугольника с такими сторонами не существует. Отдельно сообщить является ли треугольник разносторонним, равнобедренным или равносторонним.

Подсказка № 1

Треугольник может существовать только в том случае, если сумма длин любых двух его сторон больше длины третьей стороны. Проверьте это условие с помощью логического выражения.

Подсказка № 2

Внутри блока `if`, который проверяет существование треугольника, добавьте вложенные условные выражения для определения типа треугольника: равносторонний, равнобедренный или разносторонний.

Подсказка № 3

Если условие существования треугольника не выполняется, выведите сообщение, что треугольник с такими сторонами не существует.

Эталонное решение:

```
# Запрос у пользователя длин сторон треугольника
a = float(input("Введите сторону a: "))
b = float(input("Введите сторону b: "))
c = float(input("Введите сторону c: "))


# Проверка условия существования треугольника
if a + b > c and a + c > b and b + c > a:
    print("Треугольник существует.")


    # Проверка, является ли треугольник равносторонним
    if a == b == c:
        print("Треугольник равносторонний.")
    # Проверка, является ли треугольник равнобедренным
    elif a == b or b == c or a == c:
        print("Треугольник равнобедренный.")
    # Если треугольник не равносторонний и не равнобедренный, то он
разносторонний
    else:
        print("Треугольник разносторонний.")
else:
    print("Треугольник не существует.")
```

Задача 3. Простые числа

Напишите программу, которая считает количество простых чисел в заданной последовательности и выводит ответ на экран.

Простое число делится только на себя и на единицу. Последовательность задаётся при помощи вызова ввода (input) на каждой итерации цикла. Одна итерация — одно число.

Пример:

Введите количество чисел: 6.
Введите число: 4.
Введите число: 7.
Введите число: 20.
Введите число: 3.
Введите число: 11.
Введите число: 37.

Количество простых чисел в последовательности: 4.

Подсказка № 1

Чтобы определить, является ли число простым, создайте вложенный цикл. Внешний цикл будет проходить по всем числам из последовательности, а внутренний цикл будет проверять, делится ли текущее число на любые числа от 2 до его квадратного корня. Если число делится, оно не простое, и вы можете выйти из внутреннего цикла.

Подсказка № 2

Инициализируйте переменную для подсчёта количества простых чисел. Внутри внешнего цикла, если текущее число оказалось простым, увеличьте значение этой переменной на 1.

Подсказка № 3

Вложенный цикл должен проверять делимость числа от 2 до числа, меньшего текущего числа. Если число делится на любое значение, кроме 1 и самого себя, то оно не является простым.

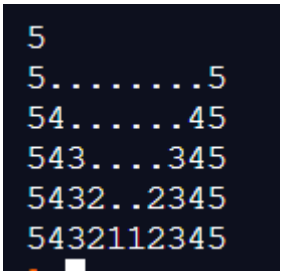
Эталонное решение:

```
# Запрос количества чисел в последовательности

n = int(input('Введите количество чисел в последовательности: '))
```

```
# Инициализация счётчика простых чисел
count = 0

# Основной цикл для ввода чисел
for i in range(n):
    # Запрос очередного числа от пользователя
    number = int(input('Введите очередное число: '))

    # Проверка числа на простоту
    if number > 1:  # Простые числа начинаются с 2
        for divider in range(2, number):
            # Если число делится на divider, оно не является простым
            if number % divider == 0:
                break
        else:
            # Если цикл завершился без break, число простое
            count += 1

# Вывод количества простых чисел в последовательности
print('Количество простых чисел:', count)
```

Задача 4. Яма

Представьте, что вы разрабатываете компьютерную игру с текстовой графикой. Вам поручили создать генератор ландшафта. Напишите программу, которая получает на вход число N и выводит на экран числа в виде ямы:



Подсказка № 1

В каждой строке ямы будут числа, точки и числа с другой стороны. Вам нужно создать структуру для генерации каждого из этих элементов в строке.

Подсказка № 2

Определите, как должна выглядеть левая часть ямы. Левая часть состоит из убывающих чисел, начиная с `depth` и уменьшаясь до значения, зависящего от текущей строки.

Подсказка № 3

Подсчитайте количество точек `.` в каждой строке. Количество точек зависит от текущей строки и будет в два раза больше разности между `depth` и номером текущей строки минус один.

Подсказка № 4

Определите, как должна выглядеть правая часть ямы. Правая часть состоит из возрастающих чисел, начиная с значения, зависящего от текущей строки, и заканчивая значением `depth`.

Эталонное решение:

```
# Запрос у пользователя глубины ямы

depth = int(input('Введите глубину ямы: '))



# Внешний цикл проходит по каждой строке ямы

for line in range(depth):

    # Генерация левой части ямы

    for left_number in range(depth, depth - line - 1, -1):

        print(left_number, end="")
```

```
    # Подсчёт количества точек
    point_count = 2 * (depth - line - 1)
    # Вывод точек
    print("." * point_count, end="")


    # Генерация правой части ямы
    for right_number in range(depth - line, depth + 1):
        print(right_number, end="")


    # Переход на новую строку после завершения текущей
    print()
```

Задача 5. Игра «Компьютер угадывает число»

Мальчик загадывает число между 1 и 100 (включительно). Компьютер может спросить у мальчика: «Твоё число равно, меньше или больше, чем число N?», где N — число, которое хочет проверить компьютер. Мальчик отвечает одним из трёх чисел: 1 — равно, 2 — больше, 3 — меньше.

Напишите программу, которая с помощью цепочки таких вопросов и ответов мальчика угадывает число.

Дополнительно: сделайте так, чтобы можно было гарантированно угадать число за семь попыток.

Подсказка № 1

Определите начальный и конечный диапазоны для возможных значений загаданного числа. Для диапазона от 1 до 100, установите переменные `start` и `finish`, где `start` равен 1, а `finish` равен 100.

Подсказка № 2

Создайте переменную `count` для подсчёта количества попыток, начиная с 1. Эта переменная поможет отслеживать, за сколько попыток компьютер угадает число.

Подсказка № 3

Используйте цикл `while`, чтобы продолжать угадывать число до тех пор, пока не получите правильный ответ. Внутри цикла вычисляйте среднее значение диапазона (`n`) с помощью `(start + finish) // 2` и спрашивайте у пользователя, является ли это число загаданным, больше или меньше.

Подсказка № 4

Обработайте ответ пользователя. Если ответ равен 1, значит число угадано, и выведите сообщение о победе. Если ответ равен 2, обновите верхний предел диапазона (`finish`) на текущее значение `n`. Если ответ равен 3, обновите нижний предел диапазона (`start`) на текущее значение `n`.

Эталонное решение:

```
# Установите начальный диапазон для возможных значений числа
start = 1
# Установите конечный диапазон для возможных значений числа
finish = 100


# Инициализируйте счётчик попыток
count = 1


# Запуск цикла угадывания числа
while True:
    # Вычисление предполагаемого числа
    n = (start + finish) // 2
    print('Загаданное число равно, меньше или больше', n)
    # Получение ответа от пользователя
    answer = int(input('1 - равно, 2 - меньше, 3 - больше '))


    # Проверка ответа пользователя
    if answer == 1:
        # Угадали число
```

```
        print('Я угадал! Ура! с', count, 'попытки')
        break
    elif answer == 2:
        # Обновление верхнего предела диапазона
        finish = n
    elif answer == 3:
        # Обновление нижнего предела диапазона
        start = n


    # Увеличение счётчика попыток
    count += 1
```