ПРОГРАММИРОВАНИЕ НА PYTHON

ДОМАШНЕЕ ЗАДАНИЕ №5

ТЕМА: ЦИКЛЫ FOR И WHILE. ФУНКЦИЯ RANGE()

ЗАДАНИЕ

ЗАДАЧА 1

На вход программе подаётся два натуральных числа n и m, каждое на отдельной строке. Напишите программу, которая печатает прямоугольник из символов '*' размерами $n \times m$.

Пример ввода:

```
3
16
```

Пример вывода:

```
****************
****************
****************
```

ЗАДАЧА 2

Напишите программу, которая считывает два целых числа m и n ($m \leq n$) и выводит в одну строку все числа от m до n включительно.

Пример ввода:

```
2
10
```

Пример вывода:

```
2 3 4 5 6 7 8 9 10
```

ЗАДАЧА 3

На вход программе подаётся натуральное число n, а затем n целых чисел, каждое на отдельной строке. Напишите программу, которая подсчитывает сумму введённых положительных чисел.

Пример ввода:

```
14
3
-5
1
10
-1
8
17
-98
31
2
-7
21
7
5
```

Пример вывода:

```
105
```

ЗАДАЧА 4

На вход программе подаётся натуральное число n. Напишите программу, которая вычисляет сумму всех его делителей.

Пример ввода:

```
50
```

Пример вывода:

```
93
```

ЗАДАЧА 5

На вход программе в цикле подаётся последовательность целых чисел делящихся на 7, каждое число на отдельной строке. При появлении любого числа, не делящегося на 7, цикл прерывается. После завершения цикла, программа должна вывести все введённые числа в одну строку.

Пример ввода:

```
7
7
14
21
13
```

Пример вывода:

```
7 7 14 21 13
```

ЗАДАЧА 6

Напишите программу, которая считывает натуральное число n и выводит первые n чисел последовательности Фибоначчи.

Последовательность Фибоначчи – это последовательность натуральных чисел, которая начинается с двух единиц, а каждое последующие число является суммой двух предыдущих: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Пример ввода 1:

```
1
```

Пример вывода 1:

```
1
```

Пример ввода 2:

```
17
```

Пример вывода 2:

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Сохраните код к каждой задаче в отдельный файл с расширением **.py**.
Упакуйте эти файлы в архив и прикрепите его в MyStat в качестве ответа.

ПОДСКАЗКИ

1. Оператор `break` осуществляет принудительный выход из цикла `for` или `while`:

```
>>> for i in range(1, 10):
...     if i % 3 == 0:
...         break
...
>>> print(i)
3
```

2. Оператор `continue` позволяет перейти к следующей итерации цикла `for` или `while`:

```
>>> for i in range(1, 10):
...     if i**2 % 10 == 1: continue
...     if i**2 % 10 != 6: print(i, end=' ')
...
2 3 5 7 8
```

3. Если мы хотим вычислить в цикле сумму некоторых значений, то необходимо заранее объявить переменную для этого:

```
>>> summ = 0
>>> for i in range(5):
...     summ += i
...
>>> print(summ)
10
```

Оператор `+=` является расширенным оператором присвоения, который перезаписывает переменную слева, прибавляя к ней значение справа. Таким образом, запись `x += 4` эквивалентна записи `x = x + 4`

4. Функция range генерирует последовательности чисел и может принимать один, два или три аргумента.

   В случае передачи одного аргумента, этим числом мы задаём верхнюю границу последовательности, нижней границей при этом всегда является ноль.

   В случае передачи двух аргументов, мы первым и вторым аргументами задаём нижнюю и верхнюю границы соответственно.

   В случае передачи трёх аргументов, первые два всё также отвечают за верхнюю и нижнюю границы диапазона последовательности, а третий аргумент изменяет шаг последовательности с единицы (значение по умолчанию) на переданное число.

```
>>> print(*range(10))
0 1 2 3 4 5 6 7 8 9
>>> print(*range(3, 7))
3 4 5 6
>>> print(*range(5, 25, 5)
5 10 15 20
>>> print(*range(20, 8, -2)
20 18 16 14 12 10
```

   Правая граница числовой последовательности никогда не включается в диапазон. Для возрастающих последовательностей – это верхняя граница, а для убывающих – нижняя.

5. У функции `print` есть скрытый аргумент, которые можно изменить. Он называется `end` и отвечает за символ, который будет напечатан в конце строки – по умолчанию это `‘\n’`, специальный символ переноса строки:

```
>>> print(“Несколько пустых строк”, end=’\n\n\n’)
Несколько пустых строк


>>> for i in range(10):
...     print(i, end=’ ‘)
... print()
...
0 1 2 3 4 5 6 7 8 9
>>>
```