Функции

Что такое функция в Python?

> **Функция** — это именованный блок кода, который выполняет какую-то полезную задачу. Её можно вызвать (запустить) в любой момент, просто указав её имя и (при необходимости) передав данные.

Зачем нужны функции?

- **Не повторять один и тот же код** много раз.
- **Упрощать программу**: сложную задачу можно разбить на части.
- **Делать код понятнее**: название функции объясняет, что она делает.

Как создать функцию?

Используется ключевое слово `def` (от английского *define* — «определить»):

```
def имя_функции(параметры):
    тело функции
    return результат  # необязательно
```

Пример:

```
def greet(name):
    return "Привет, " + name + "!"
```

Здесь:

- `greet` — имя функции,
- `name` — **параметр** (входные данные),
- `return` — команда, которая **возвращает результат** из функции.

Как вызвать функцию?

Просто напишите её имя и передайте нужные значения (**аргументы**):

```
print(greet("Аня"))  # → Привет, Аня!
```

> 💡 **Параметр** — это переменная в определении функции.
> **Аргумент** — это реальное значение, которое вы передаёте при вызове.

Что такое `return`?

- `return` отправляет результат работы функции **обратно туда, откуда её вызвали**.
- Если `return` нет, функция возвращает `None` (ничего).

Пример без `return`:

```
def say_hello():
    print("Здравствуйте!")
```

Такая функция **ничего не возвращает**, а просто выполняет действие.

Несколько параметров

Функция может принимать **сколько угодно параметров**:

```
def add(a, b):
    return a + b

print(add(3, 5))  # → 8
```

Случайные числа и выбор

Чтобы использовать случайный выбор, подключите модуль `random`:

```
import random

moods = ["весёлый", "грустный", "спокойный"]
print(random.choice(moods))  # → например: "спокойный"
```

Задания

1. Функция-предсказатель настроения

Напиши функцию `mood_predictor(name)`, которая принимает имя и "предсказывает" настроение на сегодня — весёлое, грустное, спокойное или взрывное! Используй случайный выбор (`import random, а в нем random.choice()`.

Примеры:

```
mood_predictor("Аня")     # → "Аня, сегодня у тебя настроение: ВЗРЫВНАЯ! 💥"
mood_predictor("Илья")    # → "Илья, сегодня у тебя настроение: грусное. 😌"
mood_predictor("Маша")    # → "Маша, сегодня у тебя настроение: весёлое! 😄"
```

2. Плата за такси

Представьте, что сумма за пользование услугами такси складывается из базового тарифа в размере $4,00 плюс $0,25 за каждые 140 м поездки. Напишите функцию, принимающую в качестве единственного параметра расстояние поездки в километрах и возвращающую итоговую сумму оплаты такси. В основной программе должен демонстрироваться результат вызова функции.

Примеры:

```
taxi_fare(1.4)    # → 4.25   (1400 м = 10 отрезков по 140 м → 4 + 10*0.25)
taxi_fare(0.14)   # → 4.25   (140 м → 1 отрезок)
```

> Совет: используйте `math.ceil()` для округления вверх числа отрезков.

3. Цена доставки

Интернет-магазин предоставляет услугу экспресс-доставки для части своих товаров по цене $10,95 за первый товар в заказе и $2,95 – за все последующие. Напишите функцию, принимающую в качестве единственного параметра количество товаров в заказе и возвращающую общую сумму доставки. В основной программе должны производиться запрос количества позиций в заказе у пользователя и отображаться на экране сумма доставки.

Примеры:

```
delivery_cost(1)   # → 10.95
delivery_cost(3)   # → 16.85   (10.95 + 2.95 + 2.95)
delivery_cost(5)   # → 22.75
```

4. Треугольник

Напишите функцию проверки возможности составить треугольник из трех веточек разной длину. Правило здесь простое: если длина одной сторону больше или равна сумме двух оставшихся сторон, треугольник НЕ образуется. Иначе что возможно.

Примеры:

```
is_triangle(3, 4, 5)    # → True
is_triangle(1, 1, 3)    # → False
is_triangle(10, 15, 20) # → True
```

5. Сумма

попробуйте сделать самостоятельно

Напишите функцию `sum_range(start, end)`, которая суммирует все целые числа от значения `start` до величины `end` включительно. Если пользователь задаст первое число большее чем второе, просто поменяйте их местами.

Примеры:

```
sum_range(1, 5)    # → 15   (1+2+3+4+5)
sum_range(5, 1)    # → 15   (автоматически как 1→5)
sum_range(-2, 2)   # → 0    (-2 + -1 + 0 + 1 + 2)
```

6. Функция «Генератор смешных имен»

Создай функцию `generate_funny_name(first_name)`. Она должна взять имя, перевернуть его, добавить случайный суффикс из списка (`"-ус"`, `"-тор"`, `"-ляка"`, `"-мэн"`) и случайное прилагательное (`"Неудержимый"`, `"Спящий"`, `"Восхитительный"`) в начале.

Примеры:

```
generate_funny_name("Алексей")  # → "Космический йескелА-тор"
generate_funny_name("Оля")      # → "Спящая ялО-ляка"
generate_funny_name("Коля")     # → "Неудержимый ялОК-ус"
```

> Используйте `random.choice()` и строковые операции (`[::-1]`).

7. Вес на Луне

На Луне всё не так, как на Земле! Представим, что из-за странного лунного излучения вес человека **увеличивается на фиксированную величину каждый год**.

Напишите функцию

```
moon_weight(initial_weight, yearly_gain, years)
```

которая принимает:

- `initial_weight` — начальный вес (в кг),
- `yearly_gain` — сколько килограммов добавляется **каждый год**,
- `years` — сколько лет человек проведёт на Луне,

и **возвращает итоговый вес** после этого срока.

Примеры:

```
moon_weight(30, 0.25, 15)   # → 33.75
moon_weight(50, 1.0, 5)     # → 55.0
```