



Урок 13. Python: процедура и параметры

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 13 из 34 · 45 минут · Модуль «Процедуры»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

Для трёх карточек повторён почти одинаковый код. Как исправлять оформление в одном месте и передавать разные имена и очки?

Результат урока: Выделить повторяющуюся печать в процедуру с параметрами, вызвать её несколько раз и проверить сохранность вывода.

01. Проблемная ситуация

Три карточки должны иметь строки «Участник: Лис», «Очки: 7», «---» и аналогичные для Сова с 0 очков и Кита с 12. Если формат изменится, сколько мест придётся править в трёх копиях? Предложите одну общую инструкцию с меняющимися данными.

02. Понятия и разобранный пример

5–14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Определение не равно вызову

`def` определяет подпрограмму. Её тело выполняется только при вызове, а не в момент чтения определения. В учебной терминологии процедура выполняет действие; в Python она определяется как функция, которая без явного `return` возвращает `None`.

В `def card(name, points)` имена `name` и `points` — параметры. В `card("Лис", 7)` передаются аргументы. Порядок важен: значения связываются с параметрами слева направо. Тело с отступом 4 пробела использует параметры, а вызовы без отступа находятся вне определения.

Рефакторинг меняет устройство программы при сохранении требуемого поведения. Сначала зафиксируйте ожидаемый вывод, затем вынесите повторение и сравните. Не нужно сохранять случайную ошибку старой копии. `print(card(...))` добавит строку `None` после уже напечатанной карточки; для этой процедуры нужен просто `card(...)`.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

Определяем `card` один раз и вызываем для двух участников. Первые три строки: «Участник: Лис», «Очки: 7», «---». Следующие — «Участник: Сова», «Очки: 0», «---». Ноль очков — обычный допустимый аргумент, а не признак конца.

```
def card(name, points):  
    print("Участник:", name)  
    print("Очки:", points)  
    print("---")
```

```
card("Лис", 7)  
card("Сова", 0)
```



Python: процедура и параметры

Определение	<code>def card(name, points)</code>
Вызов 1	<code>card("Лис",7)</code>
Вызов 2	<code>card("Сова",0)</code>
Результат	Один формат, разные данные

Одна процедура `card(name,points)` вызывается с разными аргументами; формат хранится в одном месте.

03. Практика 1. Проверяем материал

Печатаем карточки по образцу

Имена — вымышленные метки, сведения о реальных детях не нужны. Все исходные карточки описаны в проблемной ситуации.

Вызов	Ожидаемая вторая строка карточки
<code>card("Лис",7)</code>	Очки: 7
<code>card("Сова",0)</code>	Очки: 0
<code>card("Кит",12)</code>	Очки: 12

Сохраните `cards.py` по примеру, добавьте третий вызов `card("Кит",12)`. Предскажите все 9 строк и сравните с выводом. Затем замените разделитель `---` на `===` только в определении и проверьте все карточки. Результат — одна процедура, три вызова и протокол до/после. Критерии: нет трёх копий тела, параметры используются, ноль сохранён, правка формата действует везде.

04. Практика 2. Перенос способа

Проверяем ноль и переносим параметры

Новая процедура `line(symbol,count)` должна печатать `count` одинаковых символов без пробелов и завершать строку. Договор: `symbol` — один символ, `count` — неотрицательное целое. Используйте `for` `range` и `print(symbol,end="")`, затем отдельный `print()` после цикла. Вызовы: `line("*",4)`, `line("+",0)`, `line("#",1)`. Ожидаются строка `****`, пустая строка, строка `#`. Напишите полный `bars.py`.

05. Обсуждение и границы вывода

«Процедура всегда делает программу короче». Проверьте утверждение для одного вызова и трёх. Обоснуйте другой смысл процедуры: изменение формата и читаемость. Что произойдёт при `print(card("Лис",7))`?

06. Индивидуальная проверка

Пусть процедура `label(text)` содержит `print("[",text,")")` с отступом. Каков вывод `label("A")`? Будет ли вывод только от определения без вызова? Запишите полный вариант с двумя вызовами для А и Б. Критерии: определение/отступ, два вызова, объяснение момента выполнения.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Полный код изменённого формата ниже. На каждый вызов три строки, всего девять; разделитель === появится трижды. По 1 баллу за четыре критерия.

```
def card(name, points):
    print("Участник:", name)
    print("Очки:", points)
    print("===")

card("Лис", 7)
card("Сова", 0)
card("Кит", 12)
```

Практика 2: разбор и критерии

Ниже полный код. При count=0 цикл пуст, но завершающий print() всё равно создаёт пустую строку. Критерии: два параметра, цикл внутри процедуры, завершающий print вне цикла, три теста.

```
def line(symbol, count):
    for i in range(count):
        print(symbol, end="")
    print()

line("*", 4)
line("+", 0)
line("#", 1)
```

Обсуждение: возможный вывод

При одном вызове определение может добавить строки; главная польза — один понятный блок и единая точка правки. Внешний print напечатает None после трёх строк карточки. Это не очки и не ошибка вывода процедуры, а её возвращаемое значение.

Индивидуальная проверка: ответ и критерии

Вывод первого вызова: [A] с пробелами, поскольку print разделяет аргументы пробелом. Только определение ничего не печатает. Полный ответ ниже. 3 балла.

```
def label(text):
    print("[", text, "]")

label("A")
label("Б")
```

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, локальную папку с правом записи, Python/IDLE, распечатку ученической части PDF, бумагу и ручку. Внешние библиотеки не нужны: только стандартный Python 3.11 или новее. Все программы, входные данные, таблицы и ключи находятся в статье и PDF. Детские аккаунты, сеть и регистрация не нужны.

Уровень — начинающий, 7 класс. Предпосылки: print, for range и отступы. Работа в паре: автор прогнозирует и вводит код, проверяющий ведёт журнал; затем роли меняются. На бумаге трассируйте готовый код и входные данные; для графики используйте координатную сетку и схему. Учитель может показать заранее полученный вывод/PNG. Бумажный вариант проверяет понимание, но не выдаётся за реальный запуск учеником. Чтение, ввод и проверка включены в план. Матрица программы — материал учителя.



Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ

07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Процедуры. Процедуры с параметрами	Определяет блок и передаёт аргументы	Три карточки одним способом
Рефакторинг	Выносит повторение и сравнивает вывод	Поведение сохранено, формат меняется в одном месте

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

Официальное руководство Python 3.11: управление выполнением

<https://docs.python.org/3.11/tutorial/controlflow.html>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 17.10.2026.