



Урок 1. Python: Контракт процедуры и параметры

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 1 из 33 · 45 минут · Модуль «Документирование программы. Процедуры с параметрами»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Вызов `minutes_total(2, 15)` понятен автору. Но что означает первый аргумент и разрешено ли передать 90 минут? Договоримся об этом до запуска.

Результат урока: Описать входы, результат и ограничения подпрограммы, различить `return` и печать, проверить контракт на примерах.

01. Проблемная ситуация

0–5 минут. Два ученика прочитали `f(2,15)`: один ждёт 135 минут, другой 902 минуты. Запишите, какую информацию нужно добавить к имени и параметрам, чтобы спор исчез. Решите, считать ли 90 минут нормальным входом. В этом уроке выберем минуты внутри часа: 0...59.

02. Понятия и разобранный пример

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

Контракт: что получает и что возвращает функция

Контракт подпрограммы — описание допустимых входов, результата и заметных действий. Параметры — имена внутри определения; аргументы — конкретные значения вызова. Порядок важен: `hours` и `minutes` имеют разные единицы. В Python и процедура, и функция задаются `def`; здесь функцией называем подпрограмму с полезным возвращаемым значением.

`return` передаёт значение вызывающему коду и завершает функцию. `print` только выводит текст. Если `return` отсутствует, функция возвращает `None`. Комментарий или строка документации сами не проверяют вход: вызывающий код обязан соблюдать предусловие, либо автор отдельно программирует проверку.

Docstring — строка первой инструкцией тела функции, обычно в тройных кавычках. Она доступна через `__doc__` и `help`. Хороший контракт называет единицы и границы, а не пересказывает умножение. Здесь работаем с целыми значениями без `bool`, строк и дробей; преобразование пользовательского ввода — отдельная задача.

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

2 часа и 15 минут дают $2 \cdot 60 + 15 = 135$. Вызов `minutes_total(0,0)` даёт 0. Вызов `(23,59)` допустим и даёт 1439. `(2,90)` нарушает контракт, хотя Python вычислил бы число: отсутствие исключения не делает вызов допустимым.

```
def minutes_total(hours, minutes):  
    """Вернуть минуты для целых 0<=hours<=23, 0<=minutes<=59."""  
    return 60 * hours + minutes  
  
print(minutes_total(2, 15))  
print(minutes_total(0, 0))
```



Вход	hours=2; minutes=15
Предусловие	Целые; 0...23 и 0...59
Результат	135 минут, без печати
Проверка	23,59 → 1439; 0,0 → 0

Аргументы (2,15) → параметры hours,minutes → возвращаемое число 135; печать выполняется снаружи.

03. Практика 1. Проверяем материал

Секунды без скрытой нормализации

14–25 минут. Создайте `seconds_total(minutes, seconds)`: оба параметра целые 0...59, результат — целое число секунд, функция ничего не печатает.

Вызов <code>seconds_total(m,s)</code>	Ожидание по контракту
(0,0)	0
(1,5)	65
(59,59)	3599
(60,0); (2,60)	Нарушение предусловия

2 минуты — контракт; 4 — код и `docstring`; 3 — допустимые тесты таблицы; 2 — объяснение недопустимых. Продукт: функция и журнал «допустимость, прогноз, вывод». Критерии: единицы, две границы, `return`, отсутствие внутренней печати. Недопустимые вызовы не выдавайте за прошедшие тесты.

04. Практика 2. Перенос способа

Печать и возврат — разные обязанности

25–33 минуты. Нужна `print_duration(minutes, seconds)`, печатающая одну строку вида `2 min 5 s` и возвращающая `None`. Входы те же. За 2 минуты сравните контракт с первой практикой, за 4 напишите подпрограмму, за 2 предскажите и проверьте `print(print_duration(2,5))`. Критерии: точный формат и объяснение двух строк.

05. Обсуждение и границы вывода

33–40 минут. Один разработчик предлагает `always print`, другой — `return` и печать снаружи. Какой вариант удобнее, если результат затем надо удвоить? Приведите пример из урока и ситуацию, где процедура печати всё же уместна. Не называйте одну форму универсально лучшей.

06. Индивидуальная проверка

40–45 минут. Для `minutes_total(3,4)` назовите результат, единицу, допустимость вызова (24,0) и отличие `print` от `return`. По одному баллу за каждый ответ.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Формула $60 \cdot \text{minutes} + \text{seconds}$. Допустимые ответы 0,65,3599. Входы 60 минут или 60 секунд не входят в выбранный контракт. Можно создать другой договор с нормализацией, но нельзя молча подменить этот.

```
def seconds_total(minutes, seconds):
    """Вернуть секунды; оба аргумента целые от 0 до 59."""
    return minutes * 60 + seconds

for m, s in [(0, 0), (1, 5), (59, 59)]:
    print(seconds_total(m, s))
```

Практика 2: разбор и критерии

Внутренний вызов печатает 2 min 5 s; внешний print печатает None. Для обычного использования достаточно `print_duration(2,5)` без внешнего print. Возврат числа и печать строки — разные интерфейсы.

```
def print_duration(minutes, seconds):
    """Напечатать m min s s; целые m и s от 0 до 59."""
    print(minutes, "min", seconds, "s")

print(print_duration(2, 5))
```

Обсуждение: возможный вывод

Возвращённое 65 можно умножить на 2 и получить 130. Напечатанная строка сама не становится числом для дальнейшего вычисления. Процедура печати уместна для готового отчёта, если её контракт прямо обещает вывод. Выбор зависит от обязанности подпрограммы.

Индивидуальная проверка: ответ и критерии

184 минуты; единица — минута; (24,0) вне границы 0...23; return передаёт значение, print выводит. 4 балла. При 0–1 балле повторно подпишите параметры и единицы.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Python 3.11+ и IDLE или редактор; запуск `python lesson.py` (либо `python3 lesson.py`). Внешние библиотеки не нужны. Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 8 класс. Предпосылки: `def`, `return`, целые числа и вызов функции. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.

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
Документирование программы. Процедуры с параметрами	Разбирает пример и создаёт две подпрограммы	Исходники, прогнозы и протокол проверки
Составление и отладка программ	Сравнивает фактический результат с контрактом	Объяснение границ и самостоятельный ответ

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Python: функции и управление выполнением

<https://docs.python.org/3/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: 18.10.2026.