# Подпрограммы

Подпрограммы – это одна из очень важных разделов программирования. Трудно вообразить реальную программу без подпрограмм.

Назначение подпрограмм – это разделить большую программу на несколько более мелких программ, каждая из которых выполняет одну или несколько функций.

Что это дает программисту?

1. Структуру программы делает более логичной и наглядной, что упрощает поиск ошибок и ее корректировку.
2. Если потребуется, то многократно вызывать подпрограмму в своей программной системе на выполнение.
3. Экономить память компьютера, т.к. программный код в одном экземпляре.
4. Использовать код подпрограммы в других программных системах, если есть такая потребность.
5. Использовать подпрограммы при создании общедоступных библиотек.
6. Ну и, пожалуй, облегчает возможность распараллеливание работы, что при многопроцессорной обработке имеет важное значение, т.к. влияет на эффективность программной системы.

Начинающие программисты обычно с трудом понимают назначение подпрограмм и для чего требуется свою программу делить на подпрограммы. И только после разработки хотя бы одной программной системы начинают их активно использовать.

Итак!

**Подпрограмма** – это часть основной программы, которая оформлена таким образом, что становится возможным многократно использовать ее при выполнении основной программы.

Подпрограмма может входить в состав готовой библиотеки. В этом случае эту библиотеку требуется «заказать».

Подпрограмма – это самостоятельная «программка», которая решает какую-либо задачу.

Если программист сам разрабатывает подпрограмму, то подпрограмму требуется оформить по правилам языка программирования. Эти правила различаются в разных языках.

Любая подпрограмма имеет **имя** (по правилам программирования).

Подпрограмма может иметь **параметры.** Через параметры входные данные передаются в подпрограмму. Подпрограмма может и не иметь параметров.

Подпрограмма может возвращать результат своей работы, также через параметры, но может и не возвращать.

Рассмотрим правила оформления подпрограммы и передачи параметров в Python.

Ниже приведена простая программа, которую можно было бы написать и без подпрограмм.

Здесь подпрограммы используются исключительно в демонстрационных целях:

def get\_name():

user\_name = input("Введите свое имя: ")

return user\_name

def print\_Msg(user\_name):

print("Привет", user\_name)

def main():

user\_name = get\_name()

print\_Msg(user\_name)

main()

Любая подпрограмма начинается со слова def

Программа состоит из трех подпрограмм, имеющих названия get\_name(), print\_Msg() и main().

Подпрограмма get\_name() предлагает пользователю ввести имя, а затем возвращает значение переменной user\_name через оператор return, чтобы оно могло использоваться в других подпрограммах. Это очень важно: если не вернуть значение из подпрограммы, то значения любых переменных, созданных или измененных в этой подпрограмме, нельзя будет использовать в других точках программы.

Подпрограмма print\_Msg() выводит сообщение «Привет» и добавляет к нему имя пользователя. Переменная user\_name заключена в скобки, так как текущее значение переменной импортируется в подпрограмму для последующего использования.

Подпрограмма main() получает значение user\_name от подпрограммы get\_name(), так как оно было возвращено подпрограммой get\_name(). Затем переменная user\_ name используется в подпрограмме print\_Msg().

Последняя строка main() — это сама программа. Все, что она делает, — это запуск подпрограммы main().

Разумеется, нет никакой необходимости так хитроумно запускать очень простую программу, но здесь этот пример приводится только для того, чтобы продемонстрировать структуру подпрограмм и механизм передачи и использования переменных в подпрограммах.

Далее рассмотрим организацию подпрограмм для решения задачи хранения данных о книгах в файле в виде простой цифровой библиотеки.

Структура для хранения данных о книгах:

1. Автор - строка
2. Год издания - число
3. Название книги - строка
4. Тираж - число
5. Цена - число

Код разделен на три отдельных программы:

1. Создание нового пустого файла
2. Дозапись очередной книги
3. Печать каталога (автор и название)
4. Поиск по автору

Главная программа – это наша **main**.

Создание нового пустого файла – **create\_new\_file**

Дозапись очередной книги - **append\_book**

Печать каталога - **print\_cat**

Чтение файла и выполнение запроса – **find\_by\_author**

def main():  
 name='books.csv'

while True:  
 print("Выберите режим:")  
 print("1. Создание нового файла")  
 print("2. Дозапись книги")  
 print("3. Печать каталога")  
 print("4. Поиск по автору")  
 print("5. Выход")  
 choice = input("Введите номер режима (1-5): ")  
  
 if choice == '1':  
 create\_new\_file(name)  
 elif choice == '2':  
 append\_book(name)  
 elif choice == '3':  
 print\_cat(name)  
 elif choice == '4':  
 find\_by\_author(name)  
 elif choice == '5':  
 break  
 else:  
 print("Некорректный ввод. Попробуйте снова.")

В подпрограмме определяется имя файла books.csv, которое используется во всех других подпрограммах. Выдается список режимов и получает требуемый номер режима в переменную choice. В зависимости от режима выполняется вызов соответствующей подпрограммы. В случае ошибочного номера режима, выдается сообщение и ввод повторяется.

def create\_new\_file(name):  
 with open(name, 'w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(['Автор', 'Год выпуска', 'Название', 'Тираж', 'Цена'])  
 print("Новый файл ",name, " создан.")

Используется конструкция with для открытия файла (параметр name) в режиме записи, без конца строки (newline=''), для кириллицы (encoding='utf-8'). Объявляется объект «писатель» writer для csv файлов и пишется первая строка с названиями столбцов.

def append\_book(name):  
 while True:  
 books = []  
 books.append(input("Введите автора книги: "))  
 books.append(input("Введите год выпуска: "))  
 books.append(input("Введите название книги: "))  
 books.append(input("Введите тираж: "))  
 books.append(input("Введите цену: "))  
 with open(name, 'a', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(books)  
 print("Книга успешно добавлена в файл.")  
 con=str(input('Продолжить? (да, нет): '))  
 if con.lower() != 'да':  
 break

Организован цикл while, пока не введут «нет». В цикле объявлен пустой список books, в который добавляются данные о книге. После ввода, через with открывается файл в режиме дозаписи, создается объект writer для csv файлов и список записывается. В случае ответа «да» дозапись данных продолжается.

def print\_cat(name):  
 with open(name, 'r', encoding='utf-8') as file:  
 reader = csv.reader(file)  
 next(reader) *# Пропуск заголовка* for row in reader:  
 print(f"{row[0]} \"{row[2]}\"")  
 con=input("Посмотрели? Для продолжения нажмите Enter:")

Используется конструкция with для открытия файла (параметр name) в режиме чтения, для кириллицы (encoding='utf-8'). Создается объект writer для csv файлов и читается первая строка с названиями столбцов и выполняется переход к следующей строке (next(reader)). Далее, в цикле for читаются и печатаются авторы и название книг. Для печати используется f-строка. Для удобства, организовано «ожидание» окончания просмотра (con=input("Посмотрели? Для продолжения нажмите Enter:")).

def find\_by\_author(name):  
 author = input("Введите автора для поиска книг: ")  
 with open(name, 'r', encoding='utf-8') as file:  
 reader = csv.reader(file)  
books\_found = False  
 for row in reader:  
 if row[0] == author:  
 print("Автор:", row[0])  
 print("Год выпуска:", row[1])  
 print("Название:", row[2])  
 print("Тираж:", row[3])  
 print("Цена:", row[4])  
 print("----------------------")  
 books\_found = True  
  
 if not books\_found:  
 print("Книги автора", author, "не найдены.")

Выполняется ввод автора в переменную author. Используется конструкция with для открытия файла (параметр name) в режиме чтения, для кириллицы (encoding='utf-8'). Создается объект writer для csv файлов и читается первая строка с названиями столбцов и выполняется переход к следующей строке (next(reader)). Чтение выполняется построчно (как список) в цикле for. Каждый элемент строки (списка) имеет индекс, который используется для печати. Для отметки, что книга найдена используется признак books\_found. После окончания цикла, этот признак проверяется и выдается отрицательный результат поиска, в случае если not books\_found.

main()

Выполняется вызов подпрограммы main. Этот оператор, а также import csv единственные исполняемые команды в программе. Но часто, вместо main() пишут:

if \_\_name\_\_ == '\_\_main\_\_':

main()

В общем случае это «некрасивый», но обобщенный вариант вызова main. Можно прочитать так: «если системная переменная \_\_name\_\_ равна строке \_\_name\_\_, то выполнить оператор main(). Так рекомендуется писать если ваша программа используется в другой программе. Если же программа используется напрямую, как в нашем случае, то это необязательно.

Полный код программы:

import csv  
  
def create\_new\_file(name):  
 with open(name, 'w', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(['Автор', 'Год выпуска', 'Название', 'Тираж', 'Цена'])  
 print("Новый файл ",name, " создан.")  
  
def append\_book(name):  
 while True:  
 books = []  
 books.append(input("Введите автора книги: "))  
 books.append(input("Введите год выпуска: "))  
 books.append(input("Введите название книги: "))  
 books.append(input("Введите тираж: "))  
 books.append(input("Введите цену: "))  
 with open(name, 'a', newline='', encoding='utf-8') as file:  
 writer = csv.writer(file)  
 writer.writerow(books)  
 print("Книга успешно добавлена в файл.")  
 con=str(input('Продолжить? (да,нет): '))  
 if con.lower() != 'да':  
 break  
def print\_cat(name):  
 with open(name, 'r', encoding='utf-8') as file:  
 reader = csv.reader(file)  
 next(reader) *# Пропуск заголовка* for row in reader:  
 print(f"{row[0]} \"{row[2]}\"")  
 con=input("Посмотрели? Для продолжения нажмите Enter:")  
  
def find\_by\_author(name):  
 author = input("Введите автора для поиска книг: ")  
 with open(name, 'r', encoding='utf-8') as file:  
 reader = csv.reader(file)  
 next(reader) *# Пропуск заголовка* books\_found = False  
 for row in reader:  
 if row[0] == author:  
 print("Автор:", row[0])  
 print("Год выпуска:", row[1])  
 print("Название:", row[2])  
 print("Тираж:", row[3])  
 print("Цена:", row[4])  
 print("----------------------")  
 books\_found = True  
  
 if not books\_found:  
 print("Книги автора", author, "не найдены.")  
  
def main():  
 name='books.csv'  
 while True:  
 print("Выберите режим:")  
 print("1. Создание нового файла")  
 print("2. Дозапись книги")  
 print("3. Печать каталога")  
 print("4. Поиск по автору")  
 print("5. Выход")  
 choice = input("Введите номер режима (1-5): ")  
  
 if choice == '1':  
 create\_new\_file(name)  
 elif choice == '2':  
 append\_book(name)  
 elif choice == '3':  
 print\_cat(name)  
 elif choice == '4':  
 find\_by\_author(name)  
 elif choice == '5':  
 break  
 else:  
 print("Некорректный ввод. Попробуйте снова.")  
main()

Пока все

Как-то довольно путано расписана довольно простая суть: в питоне при импорте модуля его код выполняется целиком. Соответственно, if name == "\_\_main\_\_" это просто костыль, чтобы избежать побочных эффектов от этого (в случае если модуль содержит код, который имеет смысл выполнять только при запуске скрипта напрямую). Никакой "точки входа" или прочей магии тут нету...