

Лабораторная работа 2.8

Тема: Работа с функциями в языке Python.

Цель работы: приобретение навыков по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы


1. Создаём аккаунт в GitHub. Затем создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 RomanGorchakov ▾

Repository name *

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

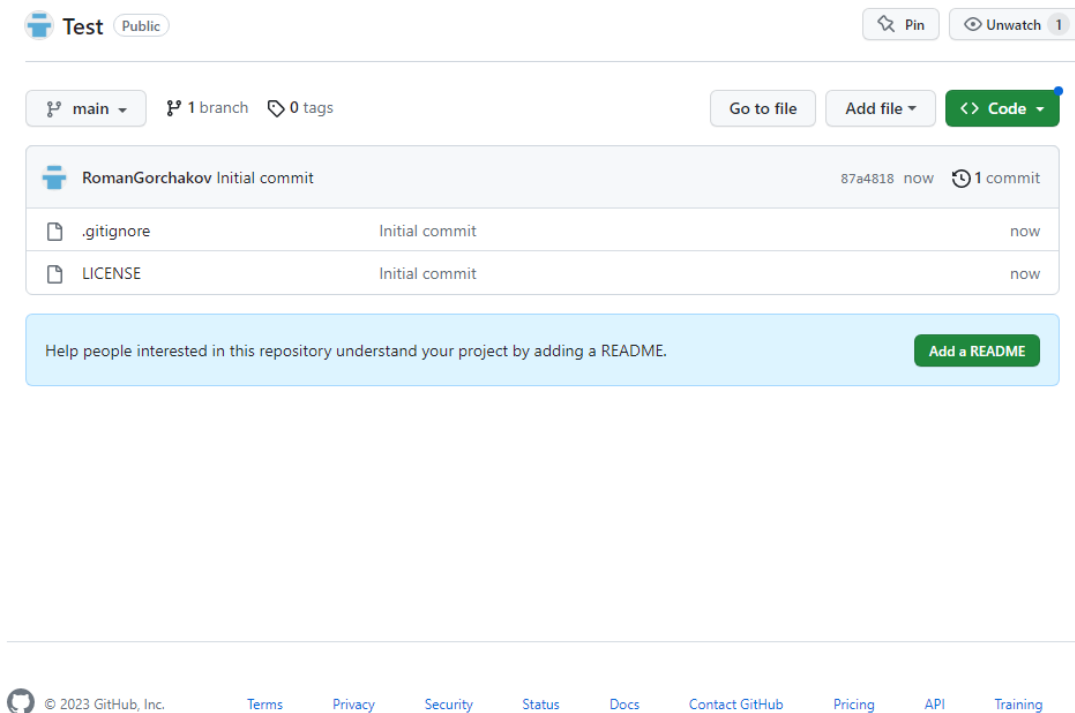
Choose a license

License: MIT License ▾


















A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository



2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».

	Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
	PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
	Plone.gitignore	Covered by global vim template	10 years ago
	Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
	Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
	PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
	Python.gitignore	Update Python.gitignore	last year
	Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
	Qt.gitignore	Remove trailing whitespace	2 years ago
	R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
	README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
	ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
	Racket.gitignore	Update Racket.gitignore	2 years ago
	Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
	Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
	RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
	Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
```

3. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».



4. После этого нужно организовать репозиторий в соответствии с моделью ветвления Git-flow. Для этого В окне «Codespace» выбираем опцию «Create codespace on main», где введём команды: «git branch develop» и «git push -u origin develop» для создания ветки разработки; «git branch feature_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix.

```
@RomanGorchakov → /workspaces/Py8 (main) $ git checkout -b develop
Switched to a new branch 'develop'
● @RomanGorchakov → /workspaces/Py8 (develop) $ git branch feature_branch
● @RomanGorchakov → /workspaces/Py8 (develop) $ git branch release/1.0.0
● @RomanGorchakov → /workspaces/Py8 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov → /workspaces/Py8 (main) $ git branch hotfix
● @RomanGorchakov → /workspaces/Py8 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov → /workspaces/Py8 (develop) $
```

5. Создаём файл «example.py», в котором нужно использовать словарь, содержащий следующие ключи: фамилия и инициалы работника; название

занимаемой должности; год поступления на работу, и написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в список, состоящий из заданных словарей; записи должны быть размещены по алфавиту;
- вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
- если таких работников нет, вывести на дисплей соответствующее сообщение.

```
>>> add
фамилия и инициалы? Svetlana Vladimirovna
Должность? Police
Год поступления? 2004
>>> add
фамилия и инициалы? Dmitry Vladimirovich
Должность? Police
Год поступления? 1990
>>> list
+-----+-----+-----+-----+
| № |          Ф.И.О.          | Должность | Год |
+-----+-----+-----+-----+
|  1 | Dmitry Vladimirovich    | Police    | 1990 |
|  2 | Svetlana Vladimirovna   | Police    | 2004 |
+-----+-----+-----+-----+
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
help - отобразить справку;
exit - завершить работу с программой.
>>> exit

...Program finished with exit code 0
Press ENTER to exit console.
```

6. Создаём файл «num.py», в котором пользователю нужно ввести слово, а программа выводит на экран, положительное ли это число или отрицательное.

```
Введите число: -100
Отрицательное

...Program finished with exit code 0
Press ENTER to exit console.
```

7. Создаём файл «cyl.py», в котором пользователю нужно ввести радиус окружности и высоту цилиндра, а программа спрашивает у пользователя, нужно ли ему найти площадь боковой поверхности цилиндра или его полную площадь.

```
Радиус окружности: 5
Высота цилиндра: 10
Что вы хотите получить: площадь боковой поверхности цилиндра или полную площадь цилиндра? Полную площадь
24674.011002723397

...Program finished with exit code 0
Press ENTER to exit console.
```

8. Создаём файл «mul.py», в котором пользователю нужно вводить числа, а программа будет их перемножать между собой до тех пор, пока очередное введённое пользователем число не станет равно нулю.

```
Введите число: 3
3
Введите число: 6
18
Введите число: 3
54
Введите число: 7
378
Введите число: 0

...Program finished with exit code 0
Press ENTER to exit console.
```

9. Создаём файл «fun.py», в котором пользователю нужно ввести строку, а программа проверяет, можно ли преобразовать эту строку в число, и выводит её, если можно.

```
50
50

...Program finished with exit code 0
Press ENTER to exit console.
```

10. Создаём файл «individual.py», в котором нужно использовать словарь, содержащий следующие ключи: название пункта назначения рейса; номер рейса; тип самолета. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию номера рейса; вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры; если таких рейсов нет, выдать на дисплей соответствующее сообщение.

```
>>> add
Название пункта назначения рейса: stavropol
>>> add
Название пункта назначения рейса: krasnodarsk
>>> add
Название пункта назначения рейса: moscow
>>> add
Название пункта назначения рейса: st. petersburg
>>> add
Название пункта назначения рейса: kazan
>>> add
Название пункта назначения рейса: volgograd
>>> list
+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолёта |
+-----+-----+-----+
| 1 | kazan | 9080 | 88 |
| 2 | krasnodarsk | 1485 | 20 |
| 3 | moscow | 1583 | 54 |
| 4 | st. petersburg | 7377 | 91 |
| 5 | stavropol | 5312 | 27 |
| 6 | volgograd | 2754 | 57 |
+-----+-----+-----+
>>> select stavropol
1: stavropol
Номер рейса: 5312
Тип самолёта: 27
>>> exit

...Program finished with exit code 0
Press ENTER to exit console.
```

11. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```
@RomanGorchakov →/workspaces/Py8 (develop) $ git add .
● @RomanGorchakov →/workspaces/Py8 (develop) $ git commit -m "Python files"
[develop c95516e] Python files
6 files changed, 307 insertions(+)
create mode 100644 cyl.py
create mode 100644 example.py
create mode 100644 fun.py
create mode 100644 individual.py
create mode 100644 mul.py
create mode 100644 num.py
● @RomanGorchakov →/workspaces/Py8 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
● @RomanGorchakov →/workspaces/Py8 (main) $ git merge develop
Updating 5bbe320..c95516e
Fast-forward
 cyl.py      | 22 ++++++
 example.py | 111 ++++++
 fun.py     | 30 ++++++
 individual.py | 99 ++++++
 mul.py     | 23 ++++++
 num.py     | 22 ++++++
6 files changed, 307 insertions(+)
create mode 100644 cyl.py
create mode 100644 example.py
create mode 100644 fun.py
create mode 100644 individual.py
create mode 100644 mul.py
create mode 100644 num.py
● @RomanGorchakov →/workspaces/Py8 (main) $ git push -u
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 3.66 KiB | 3.67 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/RomanGorchakov/Py8
 5bbe320..c95516e  main -> main
branch 'main' set up to track 'origin/main'.
○ @RomanGorchakov →/workspaces/Py8 (main) $
```

main

1 Branch

0 Tags

Go to file

Add file

Code

RomanGorchakov Python files

c95516e · now

4 Commits

.gitignore	Create .gitignore	2 weeks ago
LICENSE	Create LICENSE	2 weeks ago
README.md	Create README.md	2 weeks ago
cyl.py	Python files	now
example.py	Python files	now
fun.py	Python files	now
individual.py	Python files	now
mul.py	Python files	now
num.py	Python files	now

README

MIT license

Информация обо мне

Имя: Роман.

Фамилия: Горчаков.

Отчество: Владимирович.

Название группы: ПИЖ-6-о-22-1.

Контрольные вопросы

1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу.

2. Каково назначение операторов `def` и `return`?

В языке программирования Python функции определяются с помощью оператора `def`. Если интерпретатор Питона, выполняя тело функции, встречает `return`, то он «забирает» значение, указанное после этой команды, и «уходит» из функции.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

Локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.

5. Какие существуют способы передачи значений в функцию?

По умолчанию аргументы могут передаваться в функцию Python либо по положению, либо явно по ключевому слову.

6. Как задать значение аргументов функции по умолчанию?

Когда функция вызывается, то ей передаются аргументы. Однако на самом деле передаются не эти переменные, а их значения. Когда интерпретатор переходит к функции, чтобы начать ее исполнение, он присваивает переменным-параметрам переданные в функцию значения-аргументы.

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция.

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python – достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP – стандартизировать структуру строк документации: что они должны в себя включать, и как это написать.

9. В чем особенность однострочных и многострочных форм строк документации?

Однострочная строка документации не должна быть «подписью» параметров функции/метода. Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем интроспекции.

Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими средствами индексации, поэтому важно, чтобы она находилась на одной строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке.