

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций
Институт цифрового развития

ОТЧЁТ
по лабораторной работе №2.12

Дисциплина: «Программирование на Python»

Тема: «Декораторы функций в языке Python»

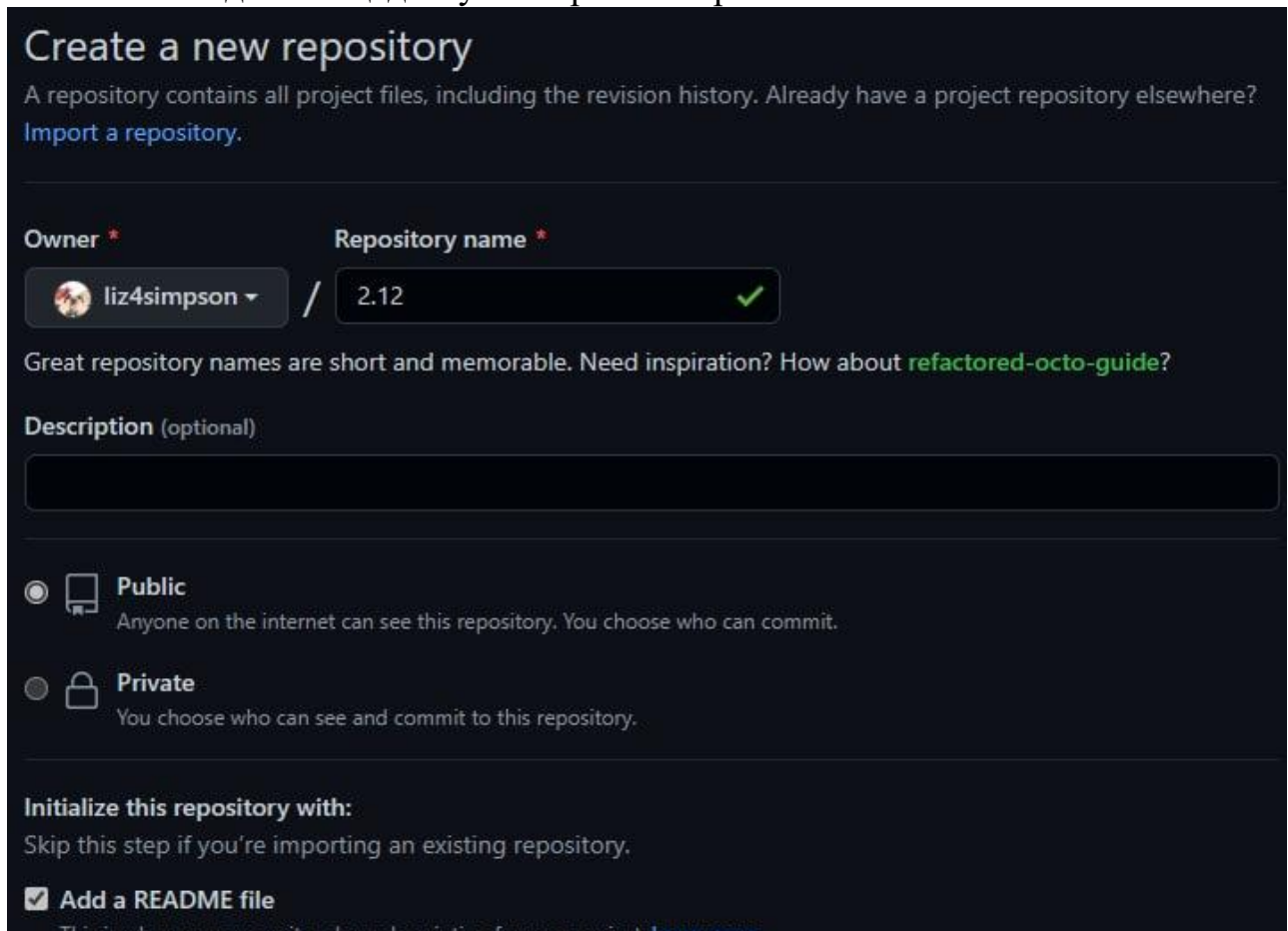
Вариант 25

Выполнила: студентка 2
курса группы ИВТ-б-о-21-1
Яковлева Елизавета
Андреевна

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

Практическая часть:


1. Создала общедоступный репозиторий на GitHub.



Create a new repository

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

Owner * **Repository name ***

 liz4simpson / 2.12 ✓

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

Description (optional)

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

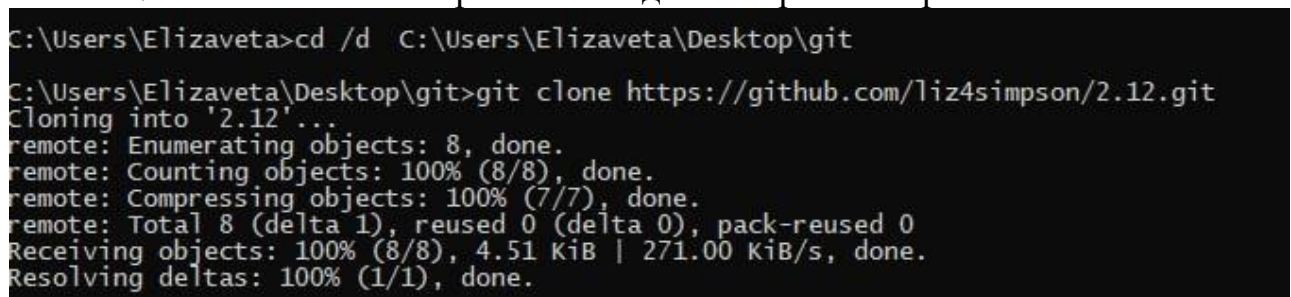
☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more](#)

Рисунок 1. Создание репозитория

2. Выполнила клонирование созданного репозитория.



```
C:\Users\Elizaveta>cd /d C:\Users\Elizaveta\Desktop\git
C:\Users\Elizaveta\Desktop\git>git clone https://github.com/liz4simpson/2.12.git
Cloning into '2.12'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), 4.51 KiB | 271.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 2. Клонирование репозитория

3. Дополнила файл .gitignore.

```
*.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
.idea/
.idea

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml
```

Рисунок 3. Изменение файла .gitignore

4. Организовала репозиторий в соответствии git-flow.

```
C:\Users\Elizaveta\Desktop\git\2.12>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Elizaveta/Desktop/git/2.12/.git/hooks]
```

Рисунок 4. Организация репозитория в соответствии с git-flow

5. Проработала примеры лабораторной работы.

```
Hello world!

Process finished with exit code 0
```

Рисунок 5. Результат выполнения примера

```
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x00000162B7795990>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

Рисунок 6. Результат выполнения примера

```
[*] Время выполнения: 1.9278357028961182 секунд.

Process finished with exit code 0
```

Рисунок 7. Результат выполнения примера

```

var a=window.innerWidth,b=window.innerHeight;if(!a||!b){var c=wind
var d=this||self,e=function(a){return a};
var g;var l=function(a,b){this.g=b===h?a:""};l.prototype.toString=
function p(a){google.timers&&google.timers.load&&google.tick&&goog
function _F_installCss(c){}
(function(){google.jl={blt:'none',chnk:0,dw:false,dwu:true,emtn:0,
Process finished with exit code 0

```

Рисунок 8. Результат выполнения примера

1. (25 вариант). Выполнила индивидуальное задание.

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def decorator(start):
5          def wrapper(func):
6              def inner(li):
7                  return func(li) + start
8              return inner
9          return wrapper
10
11
12      @decorator(start=5)
13      def s_el(li):
14          return sum([int(x) for x in li.split(' ')])
15

```

Ind x

C:\папа\2.12\Scripts\python.exe C:\Users\Elizaveta\Desktop\git\2.12\Ind.py

Введите числа через пробел: 1 2 3 4

15

Process finished with exit code 0

Рисунок 9. Вывод программы индивидуального задания

Ответы на вопросы:

1. Какие аргументы называются позиционными в Python?

Это аргументы, передаваемые в вызов в определенной последовательности (на определенных позициях), без указания их имен. Элементы объектов, поддерживающих итерирование, могут использоваться в качестве позиционных аргументов, если их распаковывать при помощи `*`.

2. Какие аргументы называются именованными в Python?

Эти аргументы, передаваемые в вызов при помощи имени (идентификатора), либо словаря с его распаковкой при помощи `**`.

3. Для чего используется оператор `*` ?

Оператор `*` чаще всего ассоциируется у людей с операцией умножения, но в Python он имеет и другой смысл.

Этот оператор позволяет «распаковывать» объекты, внутри которых хранятся некие элементы.

4. Каково назначение конструкций `*args` и `**kwargs` ?

Итак, мы знаем о том, что оператор «звёздочка» в Python способен «вытаскивать» из объектов

составляющие их элементы. Знаем мы и о том, что существует два вида параметров функций. А

именно, `*args` — это сокращение от «arguments» (аргументы), а `**kwargs` — сокращение от

«keyword arguments» (именованные аргументы).

Каждая из этих конструкций используется для распаковки аргументов соответствующего типа,

позволяя вызывать функции со списком аргументов переменной длины.

Вывод: в результате выполнения работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.