

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 №2.9

Дисциплина: «Программирование на Python»

Тема: «Рекурсия в языке Python»

Выполнил: студент 2 курса,
группы ИВТ-б-о-21-1
Богдан Александр Анатольевич

Ставрополь 2022

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

Практическая часть:

Создал общедоступный репозиторий на github

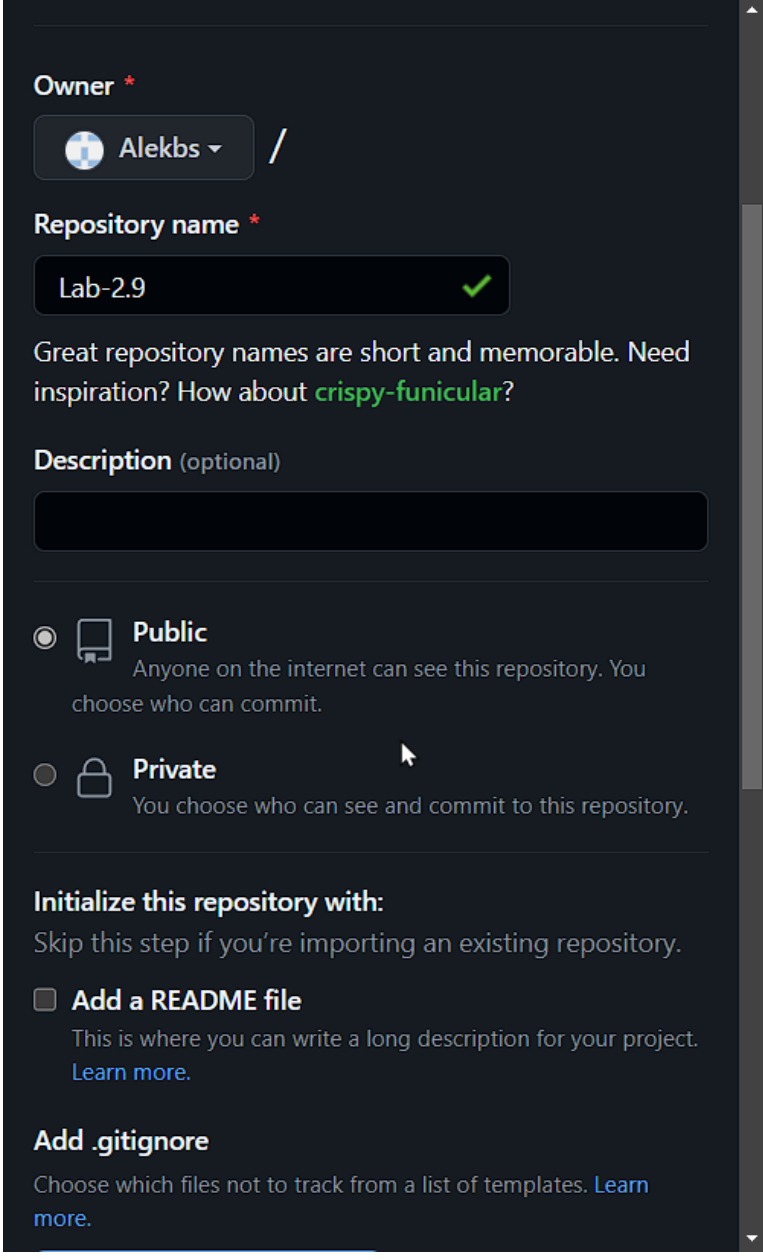
The image shows the GitHub 'Create new repository' form. At the top, the 'Owner' is set to 'Alekbs'. The 'Repository name' is 'Lab-2.9', which is marked as valid with a green checkmark. Below this, there is a tip about repository names being short and memorable, with the example 'crispy-funicular?'. The 'Description' field is empty. Under the 'Visibility' section, the 'Public' option is selected, indicating that anyone on the internet can see the repository. The 'Private' option is also visible. The 'Initialize this repository with:' section has the 'Add a README file' checkbox selected. At the bottom, there is a section for 'Add .gitignore' with a link to learn more.

Рисунок 1. Создание репозитория

Клонировал репозиторий.

```

Microsoft Windows [Version 10.0.19044.2130]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\WAR\gi>git clone https://github.com/Alekbs/Lab-2.9.git
Cloning into 'Lab-2.9'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

D:\WAR\gi>

```

Рисунок 2. Клонирование репозитория

Организовал репозиторий в соответствии с git flow

```

Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] fea
Bugfix branches? [bugfix/] bug
Release branches? [release/] rel
Hotfix branches? [hotfix/] hot
Support branches? [support/] sup
Version tag prefix? [] pre
Hooks and filters directory? [D:/WAR/gi/Lab-2.9/.git/hooks] hook

```

Рисунок 3. Организация работы в соответствии с git flow

Проработал примеры и задачи из лабораторной работы

```

task_2.py
71  if __name__ == '__main__':
72      print('Результат рекурсивного факториала:', timeit.timeit(setup=code1, number=1000))
73      print('Результат рекурсивного числа Фибоначи:', timeit.timeit(setup=code2, number=1000))
74      print('Результат итеративного факториала:', timeit.timeit(setup=code3, number=1000))
75      print('Результат итеративного числа Фибоначи:', timeit.timeit(setup=code4, number=1000))
76      print('Результат факториала с декоратором:', timeit.timeit(setup=code5, number=1000))
77      print('Результат числа Фибоначи с декоратором:', timeit.timeit(setup=code6, number=1000))
78

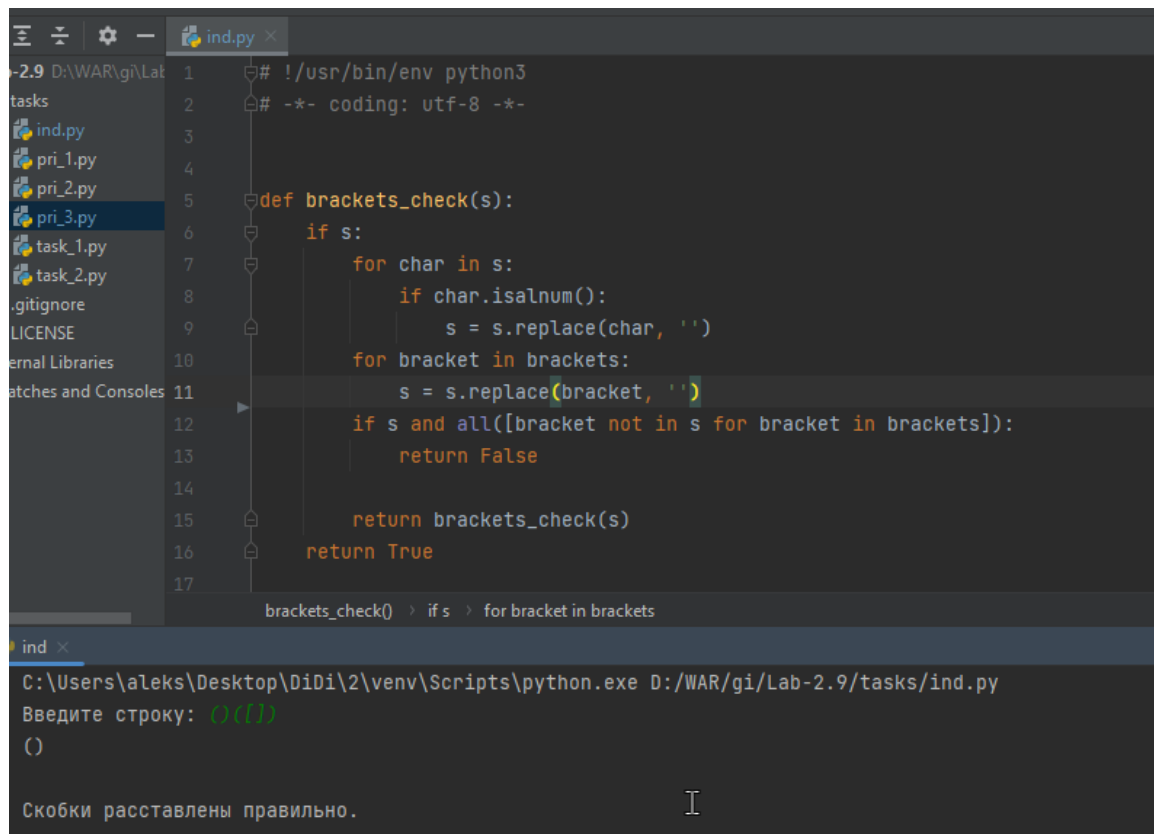
```

```

ind x task_1 x
C:\Users\aleks\Desktop\DiDi\2\venv\Scripts\python.exe D:/WAR/gi/Lab-2.9/tasks/task_1.py
Результат рекурсивного факториала: 2.560000000000145e-05
Результат рекурсивного числа Фибоначи: 2.01000000000022877e-05
Результат итеративного факториала: 2.0199999999997997e-05
Результат итеративного числа Фибоначи: 2.07000000000012375e-05
Результат факториала с декоратором: 2.1800000000001625e-05
Результат числа Фибоначи с декоратором: 2.01000000000022877e-05

```

Рисунок 4. Результат работы кода



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def brackets_check(s):
6      if s:
7          for char in s:
8              if char.isalnum():
9                  s = s.replace(char, '')
10             for bracket in brackets:
11                 s = s.replace(bracket, '')
12             if s and all([bracket not in s for bracket in brackets]):
13                 return False
14
15             return brackets_check(s)
16         return True
17
brackets_check() > if s > for bracket in brackets
```

ind ×

C:\Users\aleks\Desktop\DiDi\2\venv\Scripts\python.exe D:/WAR/gi/Lab-2.9/tasks/ind.py

Введите строку: ()()

()

Скобки расставлены правильно.

Рисунок 5. Работа индивидуального задания

Ответы на вопросы:

1. Для чего нужна рекурсия? Функция может содержать вызов других функций. В том числе процедура может вызвать саму себя. Никакого парадокса здесь нет — компьютер лишь последовательно выполняет встретившиеся ему в программе команды и, если встречается вызов процедуры, просто начинает выполнять эту функцию. Без разницы, какая функция дала команду это делать.

2. Что называется базой рекурсии? Аргументы, для которых значения функции определены (элементарные задачи).

3. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций? Стек вызовов — в теории вычислительных систем, LIFO-стек, хранящий информацию для возврата управления из подпрограмм (процедур, функций) в программу (или подпрограмму, при вложенных или рекурсивных вызовах) и/или для возврата в программу из обработчика прерывания (в том числе при переключении задач в многозадачной среде). При вызове подпрограммы или возникновении прерывания, в стек заносится адрес возврата — адрес в памяти следующей инструкции приостановленной программы и управление передается

подпрограмме или подпрограмме-обработчику. При последующем вложенном или рекурсивном вызове, прерывании подпрограммы или обработчика прерывания, в стек заносится очередной адрес возврата и т. д. При возврате из подпрограммы или обработчика прерывания, адрес возврата снимается со стека и управление передается на следующую инструкцию приостановленной (под-)программы.

4. Как получить текущее значение максимальной глубины рекурсии в языке Python? `sys.getrecursionlimit()`.

5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python? `RuntimeError: Maximum Recursion Depth Exceeded`

6. Как изменить максимальную глубину рекурсии в языке Python? Можно изменить предел глубины рекурсии с помощью вызова: `sys.setrecursionlimit(limit)`.

7. Каково назначение декоратора `lru_cache` ? `lru_cache` можно использовать для уменьшения количества лишних вычислений.

8. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов? Хвостовая рекурсия — частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Подобный вид рекурсии примечателен тем, что может быть легко заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции. Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах. В некоторых функциональных языках программирования спецификация гарантирует обязательную оптимизацию хвостовой рекурсии. Вывод: в результате выполнения работы были приобретены навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.

Вывод: Приобрели навыки по работе с рекурсивными функциями при написании программ с помощью языка программирования Python версии 3.x.