# МИНИCTEPCTBO 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 №12

Дисциплина: «Основы программной инженерии»

Тема: «Работа с множествами в языке Python»

Выполнил: студент 2 курса группы Пиж-б-о-21-1

Рязанцев Матвей Денисович

Ставрополь 2022

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

**Выполнение работы**

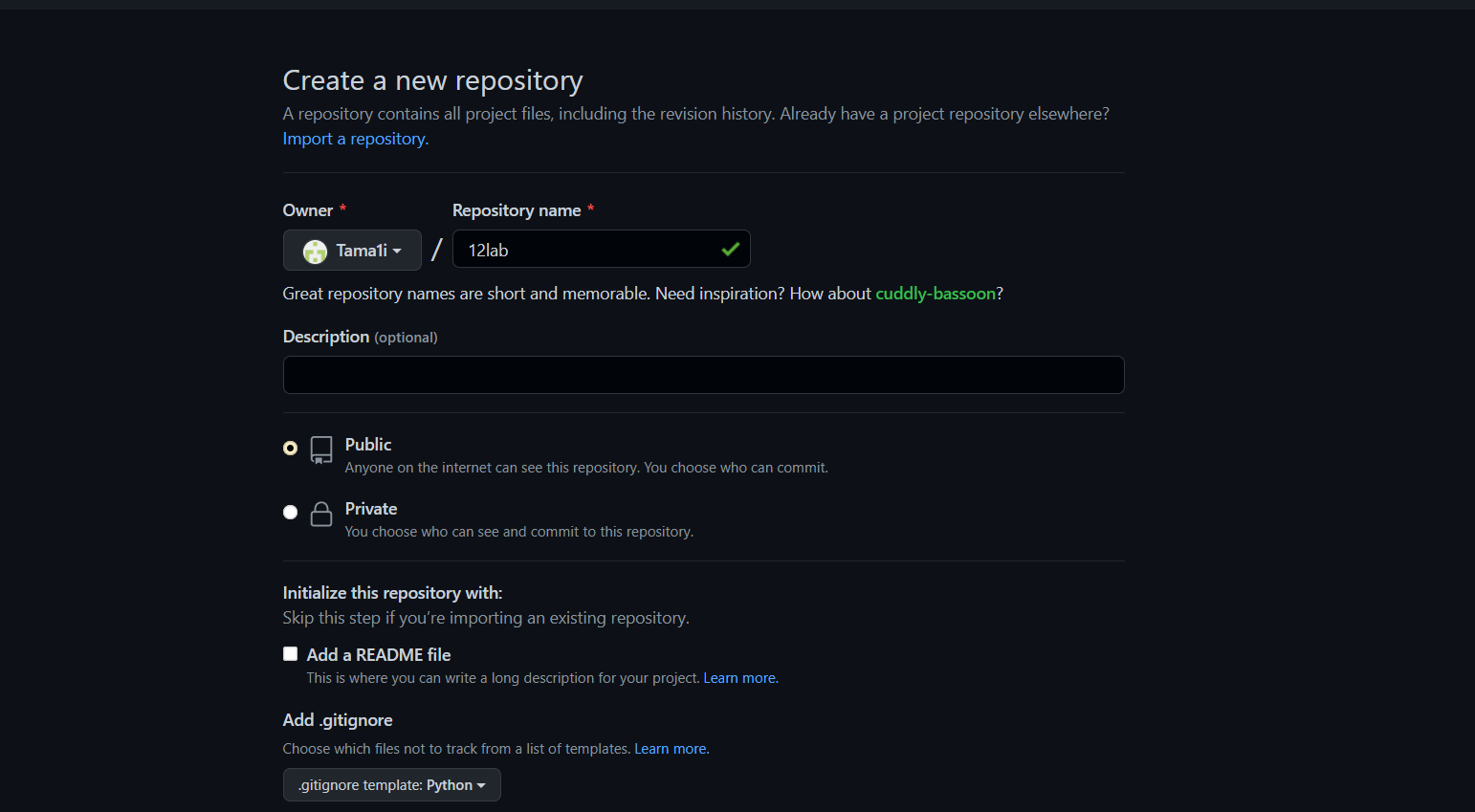


Рисунок 1 -создание репозитория

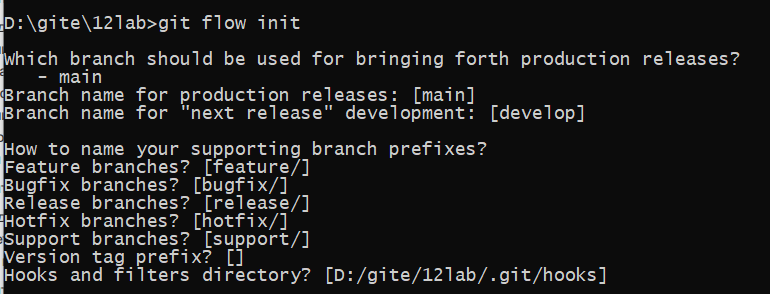


Рисунок 2 - Организация репозитория по модели ветвления git flow

Код общего задания:

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

# Эта программа показыает работу декоратора, который производит оптимизацию

# хвостового вызова. Он делает это, вызывая исключение, если оно является его

# прародителем, и перехватывает исключения, чтобы вызвать стек.

import sys

class TailRecurseException:

def \_\_init\_\_(self, args, kwargs):

self.args = args

self.kwargs = kwargs

def tail\_call\_optimized(g):

"""

Эта программа показыает работу декоратора, который производит оптимизацию

хвостового вызова. Он делает это, вызывая исключение, если оно является его

прародителем, и перехватывает исключения, чтобы подделать оптимизацию хвоста.

Эта функция не работает, если функция декоратора не использует хвостовой вызов.

"""

def func(\*args, \*\*kwargs):

f = sys.\_getframe()

if f.f\_back and f.f\_back.f\_back and f.f\_back.f\_back.f\_code == f.f\_code:

raise TailRecurseException(args, kwargs)

else:

while True:

try:

return g(\*args, \*\*kwargs)

except TailRecurseException, e:

args = e.args

kwargs = e.kwargs

func.\_\_doc\_\_ = g.\_\_doc\_\_

return func

Индивидуальное задание

Код программы:

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

import sys

def ap(arr):

if len(arr)==1:

return [arr]

else:

a=arr[0] #берем 1 "болт, сотню кротов, два ведра селикона, 2 стопки картона, 12 шестеренок,

# 3 крышки от пива и шутку админа, охапку дров - французский танк готов"

p=ap(arr[1:])

r=[]

for k in p: #цикл для пребора начального числа

for i in range(len(k)):#цикл для перебора всех позиций 1

t=k[0:i]+[a]+k[i:] #делаем разрез куда вставляем 1

r.append(t) #добавляем новую комбинацию в список комбинаций

r.append(k+[a])

return r

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

n=int(input("n="))

print(ap([i for i in range(1,n+1)]))

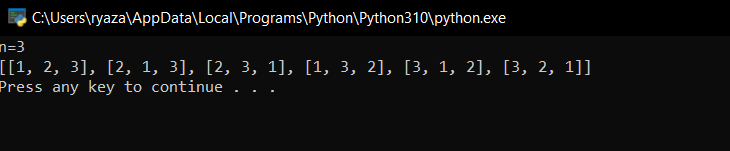


Рисунок 6 – результат работы программы идз

**Контрольные вопросы**

1. Рекурсия существенно сокращает объем кода и входит во многие

встроенные функции языков.

2. База рекурсии – это тривиальный случай, при котором решение задачи

очевидно, то есть не требуется обращение функции к себе.

3. Компьютер использует стек вызовов — специальную область памяти, где

хранит данные о точках перехода между фрагментами кода.

последовательность шагов, выполняемых при вызове функции: a. Программа

сталкивается с вызовом функции. b. Создается фрейм стека, который

помещается в стек. c. Процессор переходит к точке начала выполнения

функции. d. Инструкции внутри функции начинают выполняться. После

завершения функции, выполняются следующие шаги: e. Регистры

восстанавливаются из стека вызовов. f. Фрейм стека вытягивается из стека.

Освобождается память, которая была выделена для всех локальных

переменных и аргументов. g. Обрабатывается возвращаемое значение. h. ЦП

возобновляет выполнение кода (исходя из обратного адреса).

4. Чтобы получить текущее значение максимальной глубины рекурсии

следует вызвать функцию sys.getrecursionlimit()

5. Когда предел достигнут, возникает исключение: RuntimeError: Maximum

Recursion Depth Exceeded