

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Замыкания в языке Python»

ОТЧЕТ
по лабораторной работе №14
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

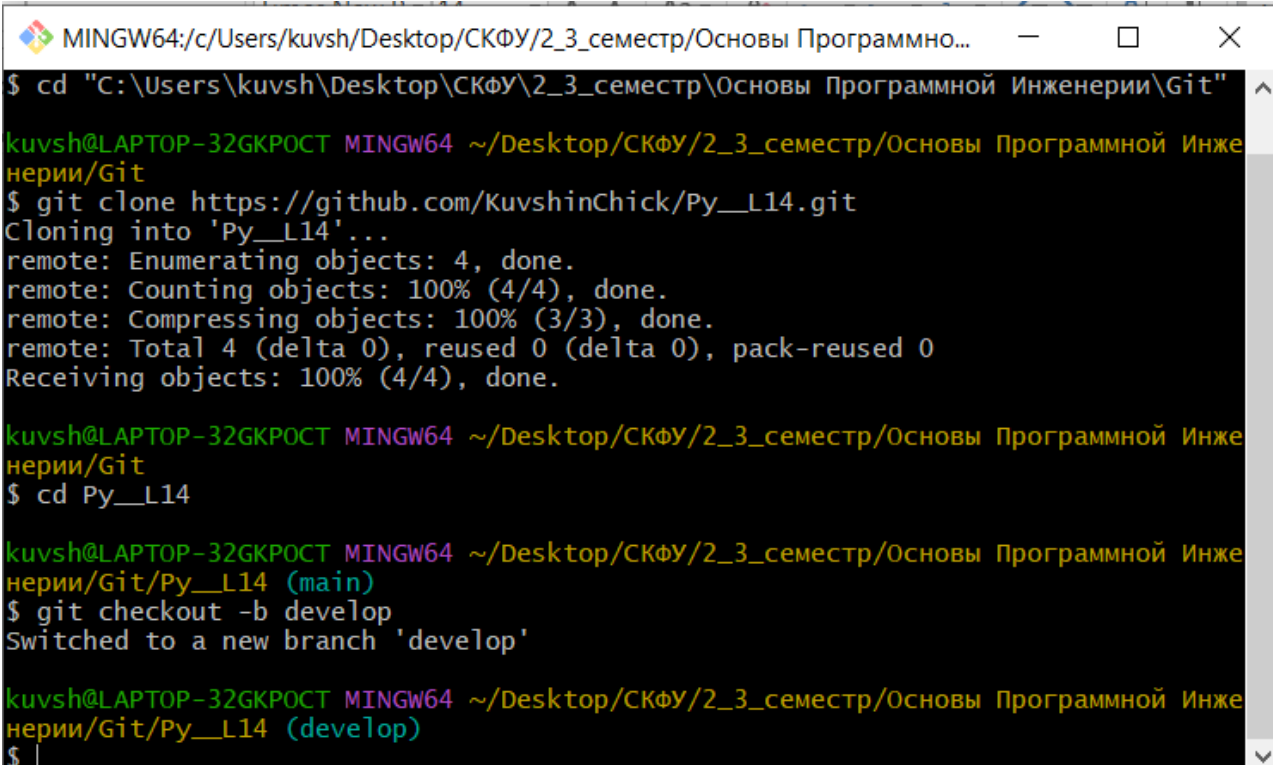
Ставрополь, 2022 г.

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

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L14.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

A screenshot of a Windows terminal window with a black background and white text. The window title is "MINGW64:/c/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программно...". The terminal shows the following commands and output:

```
$ cd "C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L14.git
Cloning into 'Py__L14'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L14
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L14 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L14 (develop)
$
```

Рисунок 14.1 – Клонирование репозитория и создание ветки develop

```

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L14 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L14 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py__L14 (develop)
$ git commit -m "modified .gitignore & readme"
[develop badef88] modified .gitignore & readme
 2 files changed, 132 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore

```

Рисунок 14.2 – Обновление .gitignore и readme

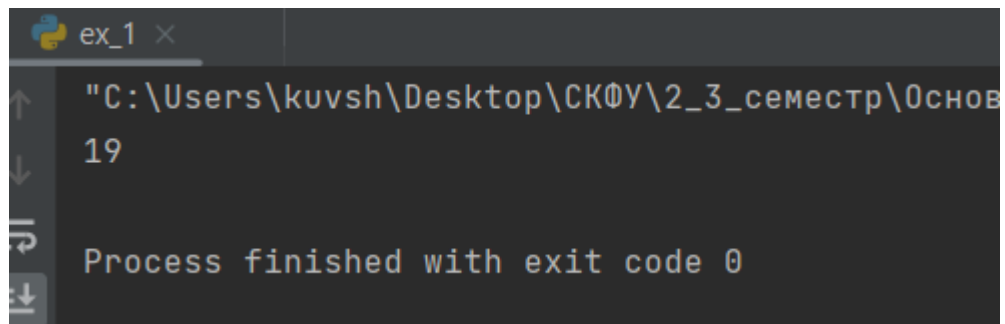
6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы.

```

ex_1.py x
1  ▶  #!/usr/bin/env python3
2  #  -*- coding: utf-8 -*-
3
4
5  def fun1(a):
6      x = a * 3
7
8      def fun2(b):
9          nonlocal x
10         return b + x
11     return fun2
12
13
14  ▶  if __name__ == '__main__':
15      test_fun = fun1(4)
16      print(test_fun(7))
17

```

Рисунок 14.3 – Код программы – примера

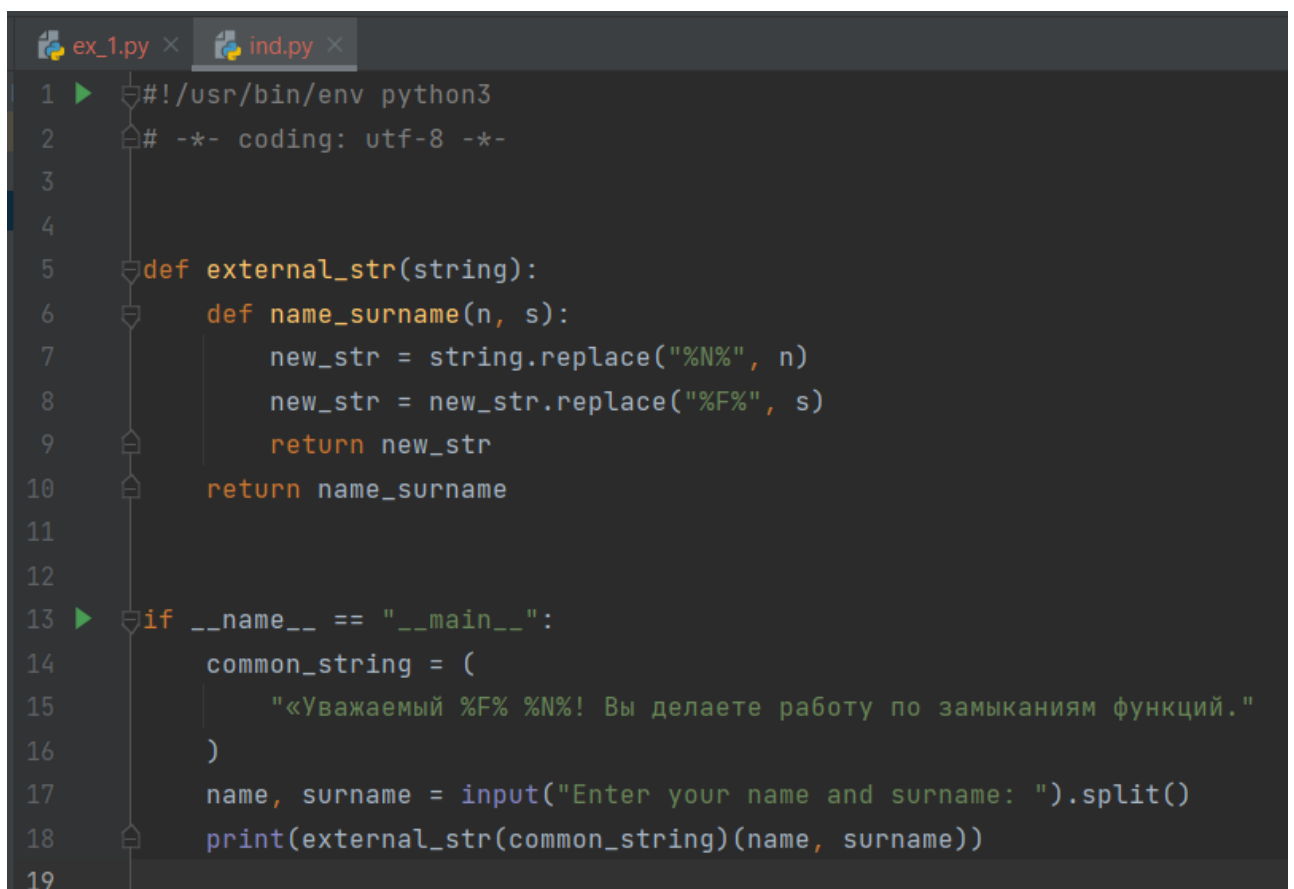


```
ex_1 x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основ
19
Process finished with exit code 0
```

Рисунок 14.4 – Результат работы программы – примера

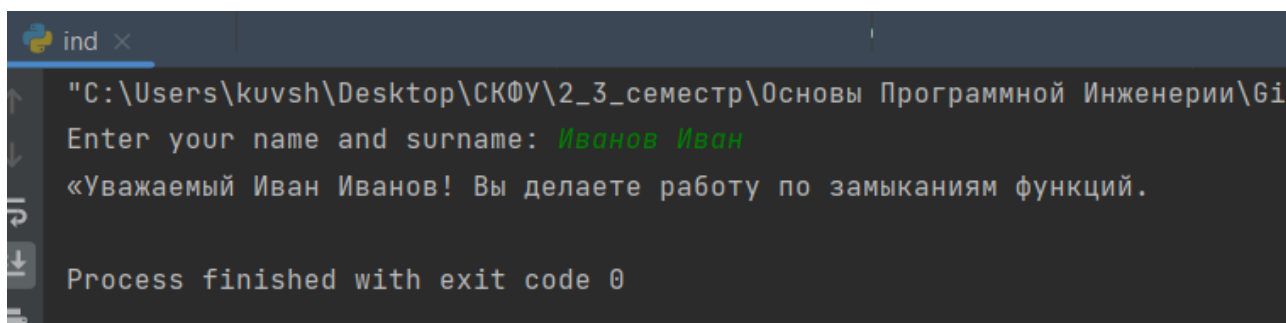
8. Выполнить индивидуальное задание.

5. Используя замыкания функций, объявите внутреннюю функцию, которая принимает в качестве параметров фамилию и имя, а затем, заносит в шаблон эти данные. Сам шаблон – это строка, которая передается внешней функции и, например, может иметь такой вид: «Уважаемый %F%, %N%! Вы делаете работу по замыканиям функций.» Здесь %F% - это фрагмент куда нужно подставить фамилию, а %N% - фрагмент, куда нужно подставить имя. (Шаблон может быть и другим, вы это определяете сами). Здесь важно, чтобы внутренняя функция умела подставлять данные в шаблон, формировать новую строку и возвращать результат. Вызовите внутреннюю функцию замыкания и отобразите на экране результат ее работы.



```
ex_1.py x ind.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 def external_str(string):
6     def name_surname(n, s):
7         new_str = string.replace("%N%", n)
8         new_str = new_str.replace("%F%", s)
9         return new_str
10    return name_surname
11
12
13 if __name__ == "__main__":
14     common_string = (
15         "«Уважаемый %F% %N%! Вы делаете работу по замыканиям функций.»
16     )
17     name, surname = input("Enter your name and surname: ").split()
18     print(external_str(common_string)(name, surname))
19
```

Рисунок 142.5 – Код программы



```
ind x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Gi
Enter your name and surname: Иванов Иван
«Уважаемый Иван Иванов! Вы делаете работу по замыканиям функций.
Process finished with exit code 0
```

Рисунок 14.6 – Результат работы программы

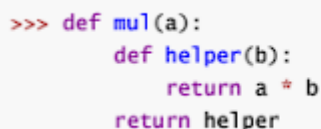
9. Зафиксируйте изменения в репозитории.
10. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
11. Выполните слияние ветки для разработки с веткой master/main.
12. Отправьте сделанные изменения на сервер GitHub.
13. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Контрольные вопросы

1. Что такое замыкание?

замыкание (closure) в программировании — это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся ее параметрами.

2. Как реализованы замыкания в языке программирования Python?



```
>>> def mul(a):
    def helper(b):
        return a * b
    return helper
```

3. Что подразумевает под собой область видимости Local?

Область видимости Local Эту область видимости имеют переменные, которые создаются и используются внутри функций.

4. Что подразумевает под собой область видимости Enclosing?

Область видимости Enclosing Суть данной области видимости в том, что внутри функции могут быть вложенные функции и локальные переменные, так

вот локальная переменная функции для ее вложенной функции находится в enclosing области видимости.

5. Что подразумевает под собой область видимости Global?

Область видимости Global Переменные области видимости global – это глобальные переменные уровня модуля (модуль – это файл с расширением .py).

6. Что подразумевает под собой область видимости Built-in?

Область видимости Built-in Уровень Python интерпретатора. В рамках этой области видимости находятся функции open, len и т. п., также туда входят исключения. Эти сущности доступны в любом модуле Python и не требуют предварительного импорта. Built-in – это максимально широкая область видимости.

7. Как использовать замыкания в языке программирования Python?

```
>>> def fun1(a):  
    x = a * 3  
    def fun2(b):  
        nonlocal x  
        return b + x  
    return fun2  
  
>>> test_fun = fun1(4)  
  
>>> test_fun(7)  
19
```

8. Как замыкания могут быть использованы для построения иерархических данных?

Теперь перейдем с уровня математики на уровень функционального программирования. Вот как определяется “свойство замыкания” в книге “Структура и интерпретация компьютерных программ” Айбельсона Х., Сассмана Д. Д. : “В общем случае, операция комбинирования объектов данных обладает свойством замыкания в том случае, если результаты соединения объектов с помощью этой операции сами могут соединяться этой же операцией”. Это свойство позволяет строить иерархические структуры данных.