

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Модули и пакеты»

ОТЧЕТ
по лабораторной работе №16
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 _____

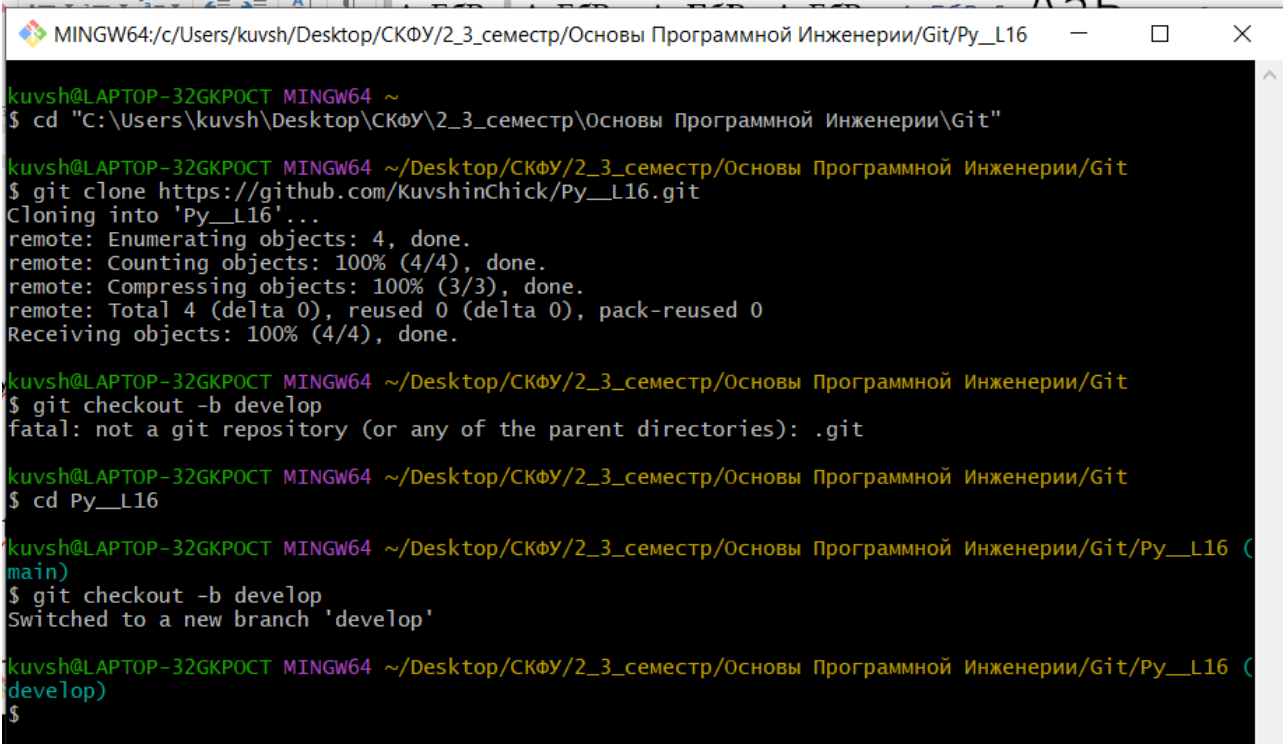
Ставрополь, 2022 г.

Цель работы: приобретение навыков по работе с модулями и пакетами языка программирования Python версии 3.x.

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L16.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 соответствие с моделью ветвления git-flow.



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L16
kuvsh@LAPTOP-32GKPOCT MINGW64 ~
$ 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__L16.git
Cloning into 'Py__L16'...
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.

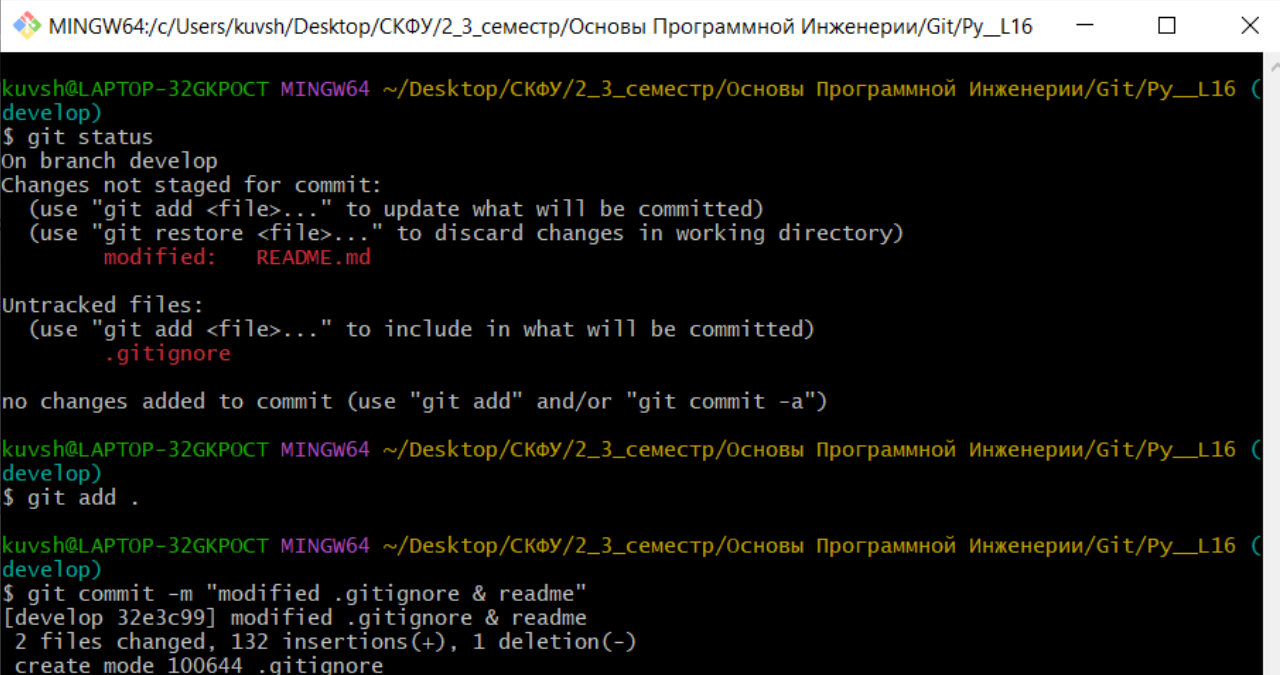
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git checkout -b develop
fatal: not a git repository (or any of the parent directories): .git

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd Py__L16

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L16 (
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__L16 (
develop)
$
```

Рисунок 16.1 – Клонирование репозитория и создание ветки develop



```
MINGW64:/c:/Users/kuvsh/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L16
kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L16 (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 Инженерии/Git/Py_L16 (develop)
$ git add .

kuvsh@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py_L16 (develop)
$ git commit -m "modified .gitignore & readme"
[develop 32e3c99] modified .gitignore & readme
 2 files changed, 132 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
```

Рисунок 16.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.

7. Выполните индивидуальные задания. Приведите в отчете скриншоты работы программ решения индивидуального задания.

Задание 1

Выполнить индивидуальное задание лабораторной работы 2.11, оформив все функции программы в виде отдельного модуля. Разработанный модуль должен быть подключен в основную программу с помощью одного из вариантов команды `import`. Номер варианта уточнить у преподавателя.

```
module.py x ind_1.py x ind_2.py x __init__.py x help.py x add.py x display.  
1 def external_str(string):  
2     def name_surname(n, s):  
3         new_str = string.replace("%N%", n)  
4         new_str = new_str.replace("%F%", s)  
5         return new_str  
6     return name_surname  
7  
8  
9 def getting_started():  
10     common_string = (  
11         "«Уважаемый %F% %N%! Вы делаете работу по замыканиям функций."  
12     )  
13     name, surname = input("Enter your name and surname: ").split()  
14     print(external_str(common_string)(name, surname))
```

```
module.py x ind_1.py x ind_2.py x __init__  
1 #!/usr/bin/env python3  
2 # -*- coding: utf-8 -*-  
3  
4 import module  
5  
6 if __name__ == '__main__':  
7     module.getting_started()  
8
```

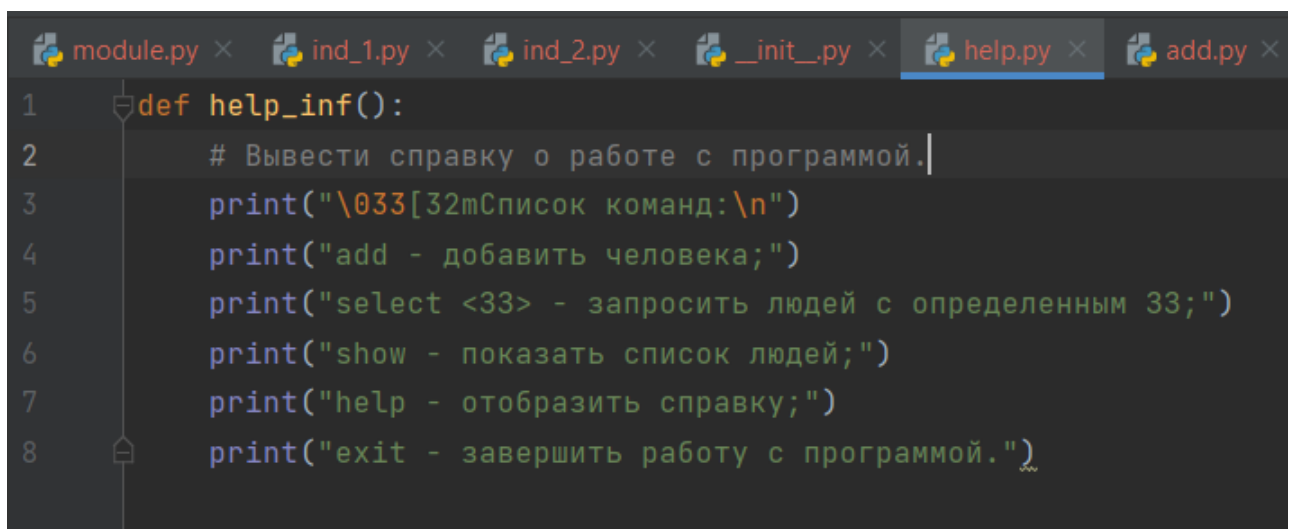
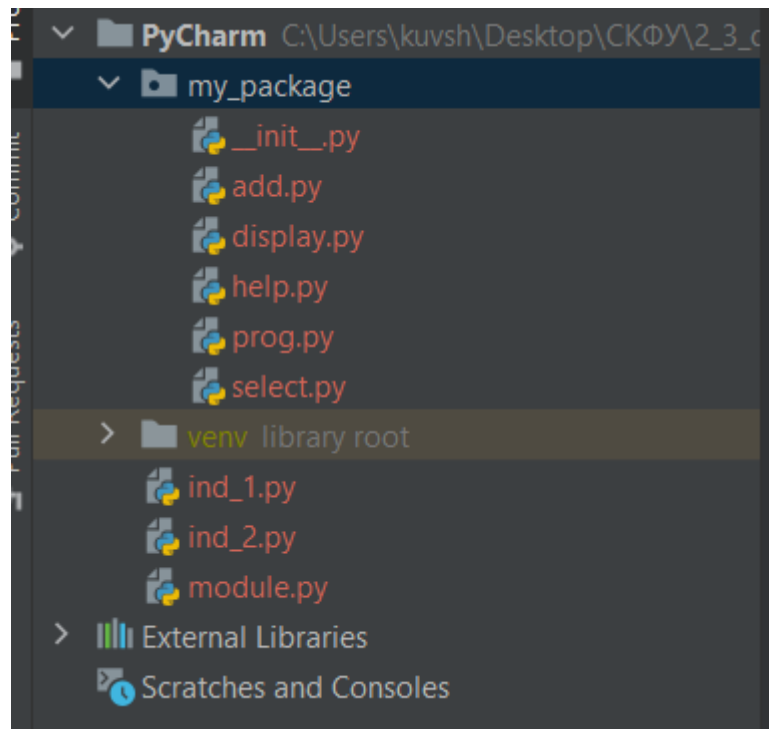
Рисунок 16.3 – Код программы

```
ind_2 x ind_1 (1) x  
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инжене  
Enter your name and surname: Ирина Кувшин  
«Уважаемый Кувшин Ирина! Вы делаете работу по замыканиям функций.  
  
Process finished with exit code 0
```

Рисунок 16.4 – Результат работы программы

Задание 2

Выполнить индивидуальное задание лабораторной работы 2.8, оформив все классы программы в виде отдельного пакета. Разработанный пакет должен быть подключен в основную программу с помощью одного из вариантов команды `import`. Настроить соответствующим образом переменную `__all__` в файле `__init__.py` пакета. Номер варианта уточнить у преподавателя.



```
module.py × ind_1.py × ind_2.py × __init__.py × help.py × add.py × display.py ×
1 import datetime
2
3
4 def add_human():
5     # Кортеж 33
6     zodiac_signs = (
7         03.21,
8         "овен",
9         04.21,
10        "телец",
11        05.22,
12        "близнецы",
13        06.22,
14        "рак",
15        07.23,
16        "лев",
17        08.23,
18        "дева",
19        09.24,
20        "весы",
21        10.24,
22        "скорпион",
```

```
module.py × ind_1.py × ind_2.py × __init__.py × help.py × add.py × display.py ×
1 def display_people(people_list):
2     # Заголовок таблицы.
3     line = (f'{"+" + "-" * 15 + "+" + "-" * 12 + "+"}'
4            f'{"-" * 15 + "+"}')
5     print(line)
6     print(f'|{'Name' : ^15}|{'Birth' : ^12}|{'Zodiac_sign' : ^15}|')
7     print(line)
8
9     # Вывести данные о всех людях.
10    for idx, man in enumerate(people_list):
11        print(
12            f'|{man.get("name", "") : ^15}'
13            f'|{man.get("birth", "") : ^12}'
14            f'|{man.get("zodiac_sign", "") : ^15}|'
15        )
16    print(line)
```

```
module.py x ind_1.py x ind_2.py x _init_.py x help.py x add.py x display.py x select.py x
1 def select_zz(com, people_list):
2     # Разбить команду на части для выделения номера года.
3     # Параметр maxsplit - int, сколько раз делить строку.
4     while True:
5         parts = com.split(' ', maxsplit=1)
6         if len(parts) == 1:
7             com = input("\033[31mEnter command "
8                 "select and Zodiac_sign: ").lower()
9             # Получить требуемый 33.
10        else:
11            break
12
13    # Инициализировать счетчик.
14    count = 0
15    zz = parts[1]
16    # Заголовок таблицы.
17    line = (f'\033[0m{"+" + "-" * 12 + "+" + "-" * 12 + "+"}'
18        f'{"-" * 15 + "+"}')
19    print(line)
20    print(
21        f"|{'Name' :^12}|{'Birth' :^12}"
```

```
module.py x ind_1.py x ind_2.py x _init_.py x help.py x add.py x display.py x select.py x prog.py x
1 from my_package import display
2 from my_package import help
3 from my_package import select
4 from my_package import add
5
6
7 def getting_started():
8     # Список людей
9     people = []
10
11    # zodiac_signs = {
12    #     'name': "овен",
13    #     'data_beg': datetime.datetime.strptime("11.11", "%d.%m"),
14    #     'data_end': "birth"
15    # }
16    # print(zodiac_signs.get("data_beg"))
17
18    # Организовать бесконечный цикл запроса команд.
19    while True:
20        # Запросить команду из терминала.
21        # На Python с помощью ANSI-код можно делать цвет, фон и т.д.
22        # \033[0m - сброс форматирования
```

```
module.py x ind_1.py x ind_2.py x _init_.py
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 from my_package import *
6
7 if __name__ == '__main__':
8     prog.getting_started()
9
```

Рисунок 16.5 – Код программы

```
ind_2 x ind_1 (1) x
"C:\Users\kuvsh\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git\Py__L16\PyCharm\venv\Script
Enter command: help
Список команд:
add - добавить человека;
select <33> - запросить людей с определенным 33;
show - показать список людей;
help - отобразить справку;
exit - завершить работу с программой.
Enter command: add
Enter name: Ирина Кувшин
Enter zodiac sign: стрелец
Enter date of birth (yyyy.mm.dd): 28.11.2003
Incorrect data format
Enter date of birth (yyyy.mm.dd): 2003.11.28
Enter command: select pak
+-----+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+-----+
Люди с заданным 33 не найдены
Enter command: select стрелец
+-----+-----+-----+
| Name | Birth | Zodiac_sign |
+-----+-----+-----+
|Ирина Кувшин| 2003.11.28 | стрелец |
+-----+-----+-----+
Enter command: |
```

Рисунок 16.6 – Результат работы программы

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

Контрольные вопросы

1. Что является модулем языка Python?

Под модулем в Python понимается файл с расширением .py. Модули предназначены для того, чтобы в них хранить часто используемые функции, классы, константы и т. п. Можно условно разделить модули и программы: программы предназначены для непосредственного запуска, а модули для

импортирования их в другие программы. Стоит заметить, что модули могут быть написаны не только на языке Python, но и на других языках (например C).

2. Какие существуют способы подключения модулей в языке Python?

Самый простой способ импортировать модуль в Python это воспользоваться конструкцией:

```
import имя_модуля
```

За один раз можно импортировать сразу несколько модулей, для этого их нужно перечислить через запятую после слова `import`:

```
import имя_модуля1, имя_модуля2
```

Если вы хотите задать псевдоним для модуля в вашей программе, можно воспользоваться вот таким синтаксисом:

```
import имя_модуля as новое_имя
```

Для импортирования нескольких функций из модуля, можно перечислить их имена через запятую

```
from имя_модуля import имя_объекта1, имя_объекта2
```

3. Что является пакетом языка Python?

Пакет в Python – это каталог, включающий в себя другие каталоги и модули, но при этом дополнительно содержащий файл `__init__.py`. Пакеты используются для формирования пространства имен, что позволяет работать с модулями через указание уровня вложенности (через точку).

4. Каково назначение файла `__init__.py` ?

В `__init__.py` файл заставляет Python рассматривать каталоги, содержащие его, как модули. Кроме того, это первый файл, загружаемый в модуль, поэтому вы можете использовать его для выполнения кода, который хотите запускать каждый раз при загрузке модуля, или для указания экспортируемых подмодулей.

5. Каково назначение переменной `__all__` файла `__init__.py`

Файл `__init__.py` может быть пустым или может содержать переменную `__all__`, хранящую список модулей, который импортируется при загрузке через конструкцию

```
from имя_пакета import *
```

Например для нашего случая содержимое `__init__.py` может быть вот таким

```
__all__ = ["simper", "compper", "annuity"]
```

```
import fincalc.simper
fv = fincalc.simper.fv(pv, i, n)
import fincalc.simper as sp
fv = sp.fv(pv, i, n)
from fincalc import simper
fv = simper.fv(pv, i, n)
```