

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

Институт цифрового развития

Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
Модули и пакеты

Выполнил:

Боженко Александр Иванович

2 курс, группа ИТС-б-о-21-1,

11.03.02 «Инфокоммуникационные
технологии и системы связи»,

направленность (профиль)

«Инфокоммуникационные системы и сети»,
очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А, канд. техн. наук, доцент
кафедры инфокоммуникаций

(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.

Краткий конспект

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*).

Самый простой способ импортировать модуль в *Python* это воспользоваться конструкцией:

```
import имя_модуля
```

Импорт и использование модуля *math*, который содержит математические функции, будет выглядеть вот так

```
import math
math.factorial(5)
120
```

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
```

При этом импортируется только конкретный объект (в нашем примере: функция *cos*), остальные функции недоступны, даже если при их вызове указать имя модуля.

```
from math import cos
```

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

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

Импортируемому объекту можно задать псевдоним.

```
from имя_модуля import имя_объекта as псевдоним_объекта
```

Если необходимо импортировать все функции, классы и т. п. из модуля, то воспользуйтесь следующей формой оператора `from ... import ...*`

```
from имя_модуля import *
```

Пакеты в 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).

Пакет *fincal* содержит в себе модули для работы с простыми процентами (`simper.py`), сложными процентами (`compper.py`) и аннуитетами (`annuity.py`).

Для использования функции из модуля работы с простыми процентами, можно использовать один из следующих вариантов:

```
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)
```

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"]
```

Примеры:

Вывод пример 1

```
120
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 1. Результат

Вывод пример 2

```
0.7071067811865476
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 2. Результат

Вывод пример 3

```
0.8660254037844386
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 3. Результат

Вывод пример 4

```
-0.9999987317275395
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 4. Результат

Вывод пример 5

```
0.5000000000000001
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 5. Результат

Вывод пример 6

```
24
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 6. Результат

Вывод пример 7

```
6.123233995736766e-17
PS C:\Users\Admin\Documents\GitHub\job-2.3>
```

Рисунок 7. Результат

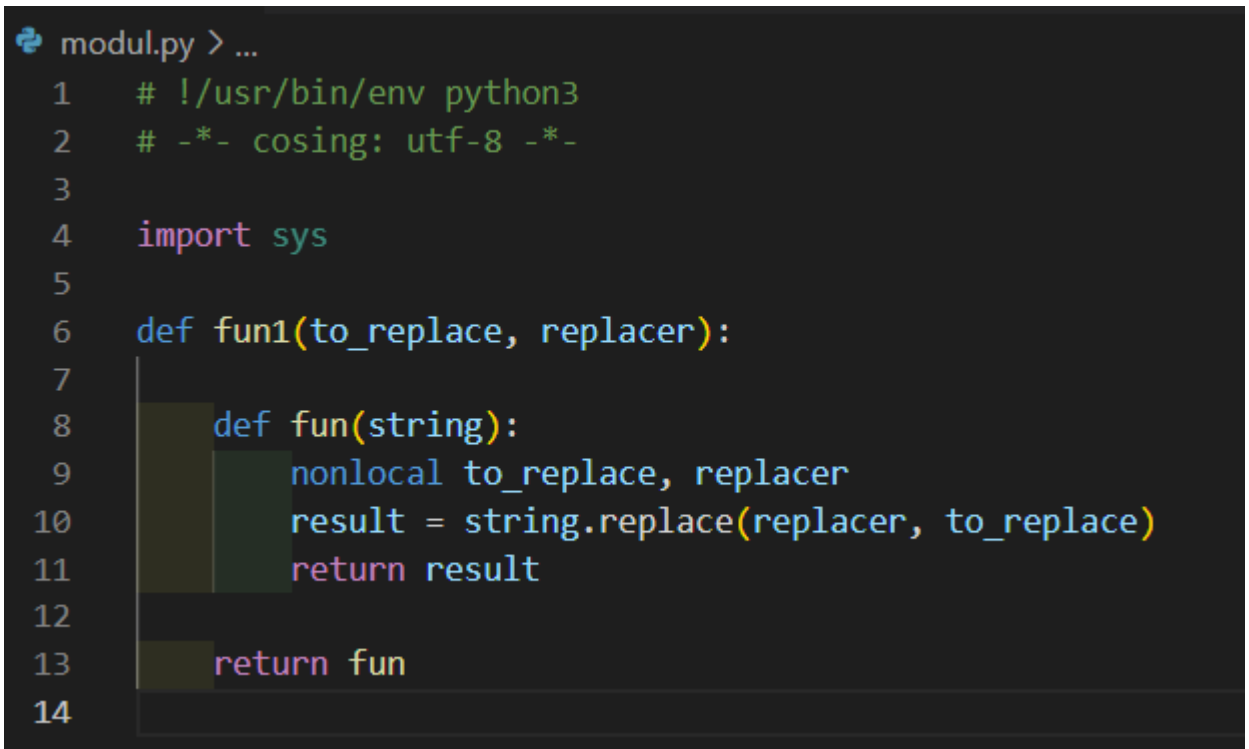
Ход работы

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`.

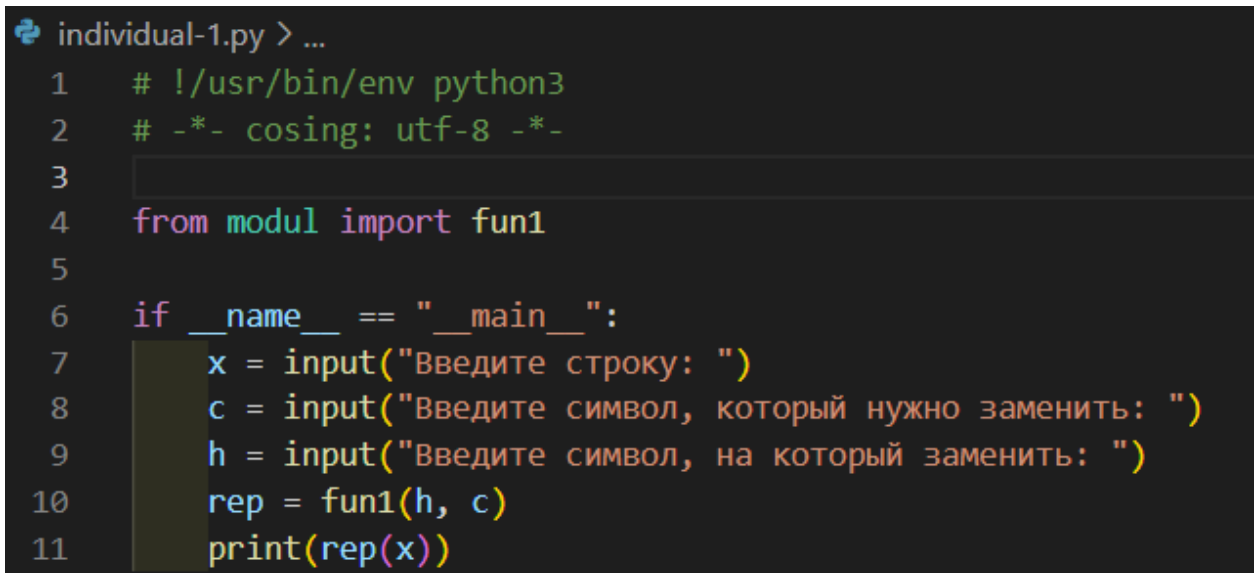
`modul.py`



```
modul.py > ...
1  # !/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  def fun1(to_replace, replacer):
7
8      def fun(string):
9          nonlocal to_replace, replacer
10         result = string.replace(replacer, to_replace)
11         return result
12
13     return fun
14
```

Рисунок 1. `modul.py`

`individual-1.py`



```
individual-1.py > ...
1  # !/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  from modul import fun1
5
6  if __name__ == "__main__":
7      x = input("Введите строку: ")
8      c = input("Введите символ, который нужно заменить: ")
9      h = input("Введите символ, на который заменить: ")
10     rep = fun1(h, c)
11     print(rep(x))
```

Рисунок 2. `individual-1.py`

Пример

```
Введите строку: 123*567
Введите символ, который нужно заменить: *
Введите символ, на который заменить: 4
1234567
```

Рисунок 3. Пример 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.

`add.py`

```
Packet > add.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import time
5
6
7  def add():
8      name = input("Название пункта назначения? ")
9      no = input("Номер поезда? ")
10     time_str = input("Введите время отправления (чч:мм)\n")
11     t = time.strptime(time_str, "%H:%M")
12     return {
13         "name": name,
14         "no": no,
15         "t": t,
16     }
```

Рисунок 4. `add.py`

`help.py`

```

Packet > 📄 help.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def help():
6      # Вывести справку о работе с программой.
7      print("Список команд:\n")
8      print("add - добавить поезд;")
9      print("list - вывести список поездов;")
10     print("select <номер> - запросить поезд по номеру;")
11     print("help - отобразить справку;")
12     print("exit - завершить работу с программой.")

```

Рисунок 5. help.py

list.py

```

Packet > 📄 list.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import time
5
6
7  def list(poezd):
8      if poezd:
9          line = "+-{}-+-{}-+-{}-+-{}-+ ".format("-" * 4, "-" * 30, "-" * 20, "-" * 13)
10         print(line)
11         print(
12             "| {:^4} | {:^30} | {:^20} | {:^13} | ".format(
13                 "No", "Название пункта", "Номер поезда", "Время"
14             )
15         )
16         print(line)
17         for idx, po in enumerate(poezd, 1):
18             print(
19                 "| {:>4} | {:<30} | {:<20} | {}{} | ".format(
20                     idx,
21                     po.get("name", ""),
22                     po.get("no", ""),
23                     time.strftime("%H:%M:%S", po.get("t", 0)),
24                     " " * 5,
25                 )
26             )
27         print(line)

```

Рисунок 6. list.py

select.py

```
Packet > select.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import time
5
6
7  def select(poezd, nom):
8      count = 0
9      for idx, po in enumerate(poezd, 1):
10         if po["no"] == str(nom):
11             print(
12                 "Название пункта: ",
13                 po["name"],
14                 "\nВремя отправления: ",
15                 time.strftime("%H:%M:%S", po["t"]),
16             )
17             count += 1
18
19     if count == 0:
20         print("Поезда с таким номером нет")
```

Рисунок 7. select.py

individual2.py


```

individual2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5  from Packet.add import add
6  from Packet.list import list
7  from Packet.select import select
8  from Packet.help import help
9
10
11 def main():
12     poezd = []
13     while True:
14         command = input(">>> ").lower()
15         if command == "exit":
16             break
17         elif command == "add":
18             po = add()
19             poezd.append(po)
20         elif command == "list":
21             list(poezd)
22         elif command.startswith("select"):
23             nom = input("Введите номер поезда: ")
24             select(poezd, nom)
25         elif command == "help":
26             help()
27         else:
28             print(f"неизвестная команда {command}", file=sys.stderr)
29
30
31 if __name__ == "__main__":
32     main()

```

Рисунок 8. individual2.py

```

>>> add
Название пункта назначения? Ставрополь
Номер поезда? 21
Введите время отправления (чч:мм)
13:22
>>> list

```

No	Название пункта	Номер поезда	Время
1	Ставрополь	21	13:22:00

Рисунок 9. Пример программы

Ответы на контрольные вопросы

1. Что является модулем языка Python?

Модули и пакеты значительно упрощают работу программиста. Классы, объекты, функции и константы, которыми приходится часто пользоваться можно упаковать в модуль, и, в дальнейшем, загружать его в свои программы при необходимости. Пакеты позволяют формировать пространства имен для работы с модулями.

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*

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.

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` ?

В `__all__` перечислены имена переменных, которые будут импортированы в область видимости модуля при вызове `from package import*`.