

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 №11
дисциплины «Основы программной инженерии»

Выполнил:
Звездин Алексей Сергеевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Ход работы

1. Я изучил теоретический материал работы

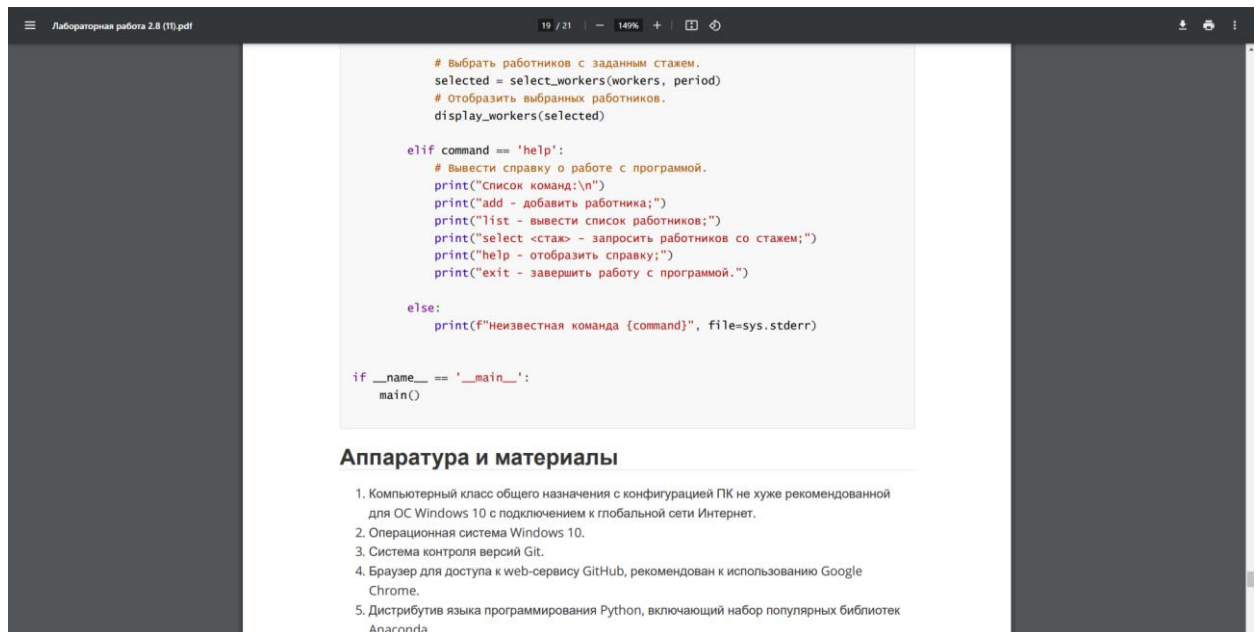


Рисунок 1.1 – Изучение материала для лабораторной работы


2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 InternetHacker1123 ▾

Repository name *

/ laba 2.8(11)

✔ Your new repository will be created as `laba-2.8-11-`.

The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

Great repository names are short and memorable. Need inspiration? How about [reimagined-carnival](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add `.gitignore`

`.gitignore` template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 2.1 – Настройка репозитория

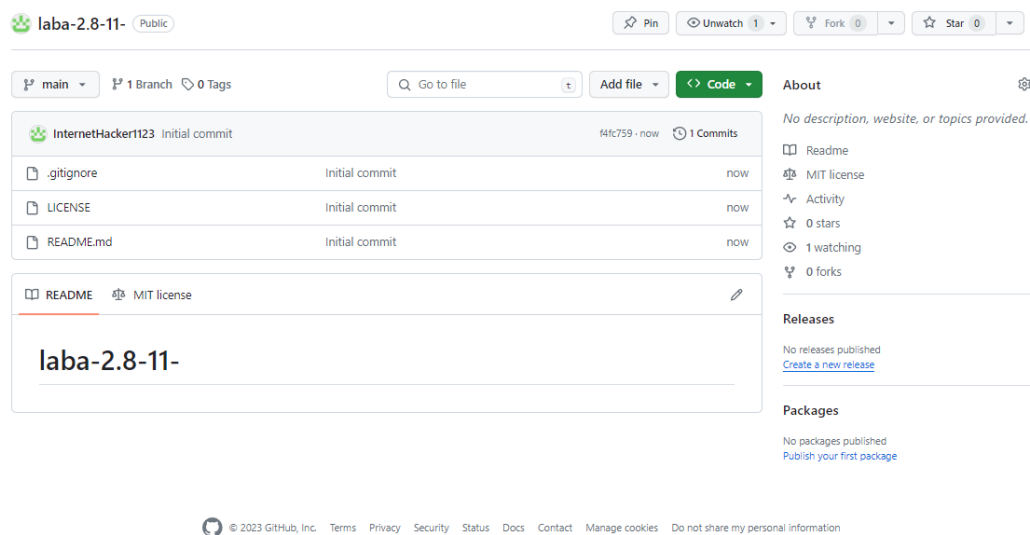


Рисунок 2.2 – Готовый репозиторий

3. Выполняю клонирование созданного репозитория

```
C:\Users\tyt\Desktop\SE>cd laba11
C:\Users\tyt\Desktop\SE\laba11>git clone https://github.com/InternetHacker1123/laba-2.8-11-.git
Cloning into 'laba-2.8-11-'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\tyt\Desktop\SE\laba11>
```

Рисунок 3.1 – Клонирование репозитория на локальный диск

4. Дополнил файл .gitignore необходимыми правилами для работы с VS Code

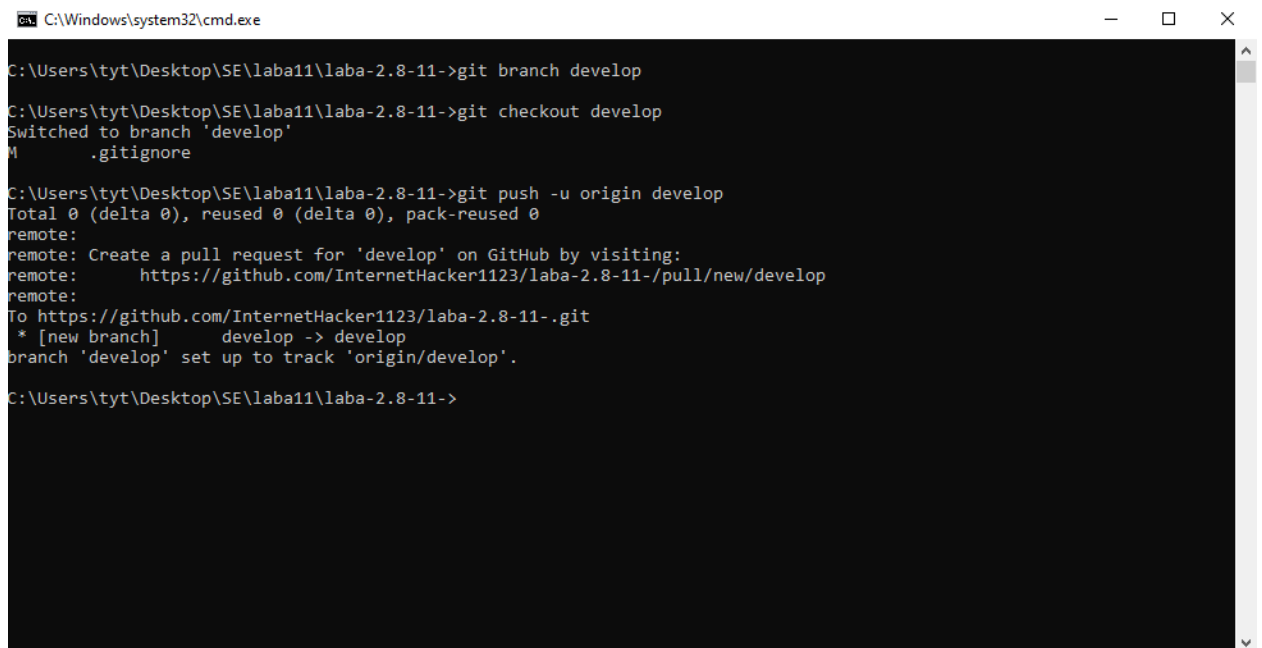
```

152 # Cython debug symbols
153 cython_debug/
154
155 .vscode/*
156 !.vscode/settings.json
157 !.vscode/tasks.json
158 !.vscode/launch.json
159 !.vscode/extensions.json
160 !.vscode/*.code-snippets
161
162 # Local History for Visual Studio Code
163 .history/
164
165 # Built Visual Studio Code Extensions
166 *.vsix
167

```

Рисунок 4.1 – .gitignore для VS Code

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow



```

C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git branch develop
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git checkout develop
Switched to branch 'develop'
M       .gitignore
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/InternetHacker1123/laba-2.8-11-/pull/new/develop
remote:
To https://github.com/InternetHacker1123/laba-2.8-11-.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->

```

Рисунок 5.1 – Создание ветки develop от ветки main

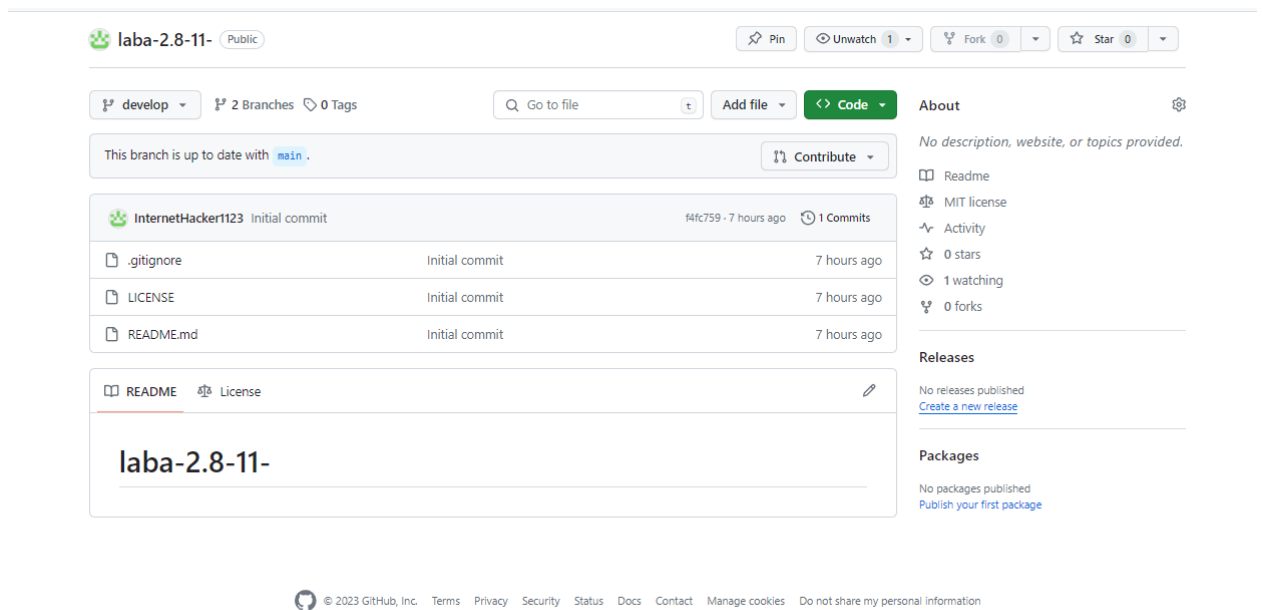


Рисунок 5.2 – Ветка develop на GitHub

6. Создал проект PyCharm в папке репозитория

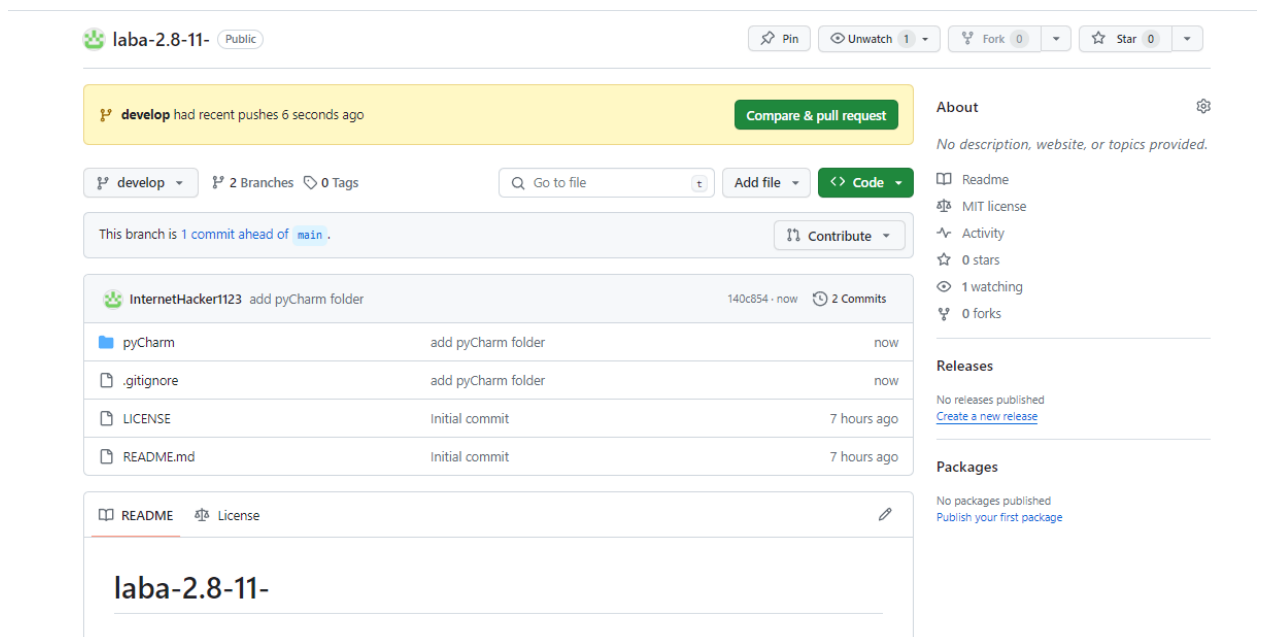


Рисунок 6.1 – Репозиторий с проектом PyCharm

7. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

```
.gitignore example1.py 1, M X
pyCharm > example1.py > display_workers
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  def get_worker():
7      """
8      Запросить данные у работника.
9      """
10     name = input("Фамилия и инициалы? ")
11     post = input("Должность? ")
12     year = int(input("Год поступления? "))
13
14     # Создать словарь.
15     return {
16         'name': name,
17         'post': post,
18         'year': year,
19     }
20
21
22 def display_workers(staff):
23     """
24     Отобразить список работников.
25     """
26     # Проверить, что список работников не пуст.
27     if staff:
28         # Заголовок таблицы.
29         line = '+-{}-+-{}-+-{}-+-{}-+'.format(
30             '-' * 4,
31             '-' * 30,
32             '-' * 20,
33             '-' * 8
34         )
35         print(line)
36         print(
37             '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
38                 "№",
39                 "Ф.И.О.",
40                 "Должность",
41                 "Год"
42             )
43         )
44         print(line)
45
46         # Вывести данные о всех сотрудниках.
47         for idx, worker in enumerate(staff, 1):
48             print(
49                 '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
```

Рисунок 7.1 – Проработка примера 1

```
C:\Windows\system32\cmd.exe
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git add .
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git commit -m"add example"
[develop 1e373c5] add example
1 file changed, 129 insertions(+)
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.75 KiB | 1.75 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/InternetHacker1123/laba-2.8-11-.git
140c854..1e373c5 develop -> develop
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->
```

Рисунок 7.5 – Фиксирование изменений в репозитории

8. Решил следующую задачу: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное".

```
pyCharm > task1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  def test():
7      number = int(input("Введите целое число: "))
8      if number == 0:
9          print("Не ноль!", file=sys.stderr)
10         exit(1)
11     if number > 0:
12         positive()
13     elif number < 0:
14         negative()
15
16
17
18 def positive():
19     print("Положительное")
20
21
22 def negative():
23     print("Отрицательное")
24
25
26 if __name__ == '__main__':
27     test()
```

Рисунок 8.1 – Код программы task1.py

9. Зафиксировал сделанные изменения в репозитории


```
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git commit -m"add task1"
[develop 15261dd] add task1
 4 files changed, 28 insertions(+)
 create mode 100644 pyCharm/task1.py
 create mode 100644 pyCharm/task2.py
 create mode 100644 pyCharm/task3.py
```

Рисунок 9.1 – Коммит файлов в репозитории git

10. Решил следующую задачу: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.

```

pyCharm > task2.py > cylinder
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import math
5
6
7  def cylinder():
8      # Определить функцию для нахождения площади круга.
9      def circle(radius):
10         return math.pi * radius ** 2
11
12     # Ввести радиус и высоту цилиндра.
13     radius = float(input("Введите радиус цилиндра: "))
14     visota = float(input("Введите высоту цилиндра: "))
15
16     # Посчитать площади боковой и всей поверхностей.
17     ploshad_boka = 2 * math.pi * radius * visota
18     vsya_ploshad = (ploshad_boka + 2 * circle(radius))
19
20     # Узнать хочет ли получить пользователь всю площадь.
21     zapros = input("Вам нужна полная площадь цилиндра?")
22
23     if zapros.lower() != "да":
24         print(f"Площадь боковой поверхности цилиндра: "
25               f"{ploshad_boka}")
26     else:
27         print(f"Полная площадь цилиндра: "
28               f"{vsya_ploshad}")
29
30
31 if __name__ == '__main__':
32     cylinder()

```

Рисунок 10.1 – Код программы task2.py

11. Зафиксировал сделанные изменения в репозитории.

```

C:\TestGit\WeArePythonistsAroudPeople>git add PyCharm
C:\TestGit\WeArePythonistsAroudPeople>git commit -m"add task 10
[develop d6afb5d] add task 10
1 file changed, 41 insertions(+)

```

Рисунок 11.1 – Коммит файлов в репозитории git

12. Решил следующую задачу: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0.

Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.

```
pyCharm > task3.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def multiply():
5      mult = 1
6      number = int(input("Введите число: "))
7
8      while number != 0:
9          mult *= number
10         number = int(input("Введите число: "))
11
12     print(mult)
13
14
15 if __name__ == '__main__':
16     multiply()
```

Рисунок 12.1 – Код программы task3.py

13. Зафиксировал сделанные изменения в репозитории.

```
C:\TestGit\WeArePythonistsA proudPeople>git add PyCharm
C:\TestGit\WeArePythonistsA proudPeople>git commit -m"add task 12"
[develop 79a9273] add task 12
 2 files changed, 22 insertions(+), 1 deletion(-)
```

Рисунок 13.1 – Коммит файлов в репозитории git

14. Решил следующую задачу: напишите программу, в которой определены следующие четыре функции:

1. Функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2. Функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.

3. Функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4. Функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

```
pyCharm > task4.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def get_input():
6      line = input("Введите любую строку: ")
7      return line
8
9
10 def test_input(variable):
11     try:
12         int(variable)
13         return True
14     except ValueError:
15         return False
16
17
18 def str_to_int(number):
19     string_number = int(number)
20     return string_number
21
22
23 def print_int(value):
24     print(value)
25
26
27 if __name__ == '__main__':
28     variable = get_input()
29
30     if test_input(variable):
31         return_value = str_to_int(variable)
32         print_int(return_value)
33     else:
34         print("Вы ввели не число")
```

Рисунок 14.1 – Код программы `task4.py`

15. Зафиксировал сделанные изменения в репозитории.

```
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git add .  
  
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git commit -m"add task4"  
[develop d63c0cd] add task4  
1 file changed, 34 insertions(+)  
create mode 100644 pyCharm/task4.py  
  
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->
```

Рисунок 15.1 – Коммит файлов в репозитории git

16. Привел в отчете скриншоты результатов выполнения примера при различных исходных данных, вводимых с клавиатуры.

```
PS C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11-> & C:/Users/tyt/AppData/Local/Programs/Python/Python311/python.exe c:/Users/tyt/Desktop/SE/laba11/laba-2.8-11-/pyCharm/example1.py  
>>> add  
Фамилия и инициалы? Ху Леша Таович  
Должность? Главный геншинер  
Год поступления? 2022  
>>> list  
+-----+-----+-----+-----+  
| № | Ф.И.О. | Должность | Год |  
+-----+-----+-----+-----+  
| 1 | Ху Леша Таович | Главный геншинер | 2022 |  
+-----+-----+-----+-----+  
>>> █
```

Рисунок 16.1 – Результат программы example1.py

```
PS C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11-> & C:/Users/tyt/AppData/Local/Programs/Python/Python311/python.exe c:/Users/tyt/Desktop/SE/laba11/laba-2.8-11-/pyCharm/example1.py  
>>> add  
Фамилия и инициалы? Роман Сенсей Воронкин  
Должность? Питон Мастер  
Год поступления? 2023  
>>>  
Неизвестная команда  
>>> list  
+-----+-----+-----+-----+  
| № | Ф.И.О. | Должность | Год |  
+-----+-----+-----+-----+  
| 1 | Роман Сенсей Воронкин | Питон Мастер | 2023 |  
+-----+-----+-----+-----+  
>>> █
```

Рисунок 16.2 – Результат программы example1.py

17. Приведите в отчете скриншоты работы программ решения индивидуального задания.

```
pyCharm > individual.py > main
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def add(sp):
5      punkt_naznachenia = input("Пункт назначения поезда: ")
6      train_number = input("Номер поезда: ")
7      time_otpravlenia = input("Время отправления: ")
8
9      dictionary = {
10         'Пункт назначения ': punkt_naznachenia,
11         'Номер поезда: ': train_number,
12         'Время отправления:': time_otpravlenia
13     }
14
15     sp.append(dictionary)
16     sp = sorted(sp, key=lambda x: x['Номер поезда: '])
17
18  def choose(sp):
19     inp = input("Введите номер поезда: ")
20     for d in sp:
21         if inp in d.values():
22             print(d)
23         else:
24             print('Поезда с таким номером нет')
25
26  def get_list(sp):
27     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
28         '-' * 4,
29         '-' * 30,
30         '-' * 20,
31         '-' * 20
32     )
33     print(line)
34     print(
35         '| {:^4} | {:^30} | {:^20} | {:^20} |'.format(
36             "№",
37             "Пункт назначения",
38             "Номер поезда",
39             "Время отправления"
40         )
41     )
42     print(line)
43     for idx, train in enumerate(sp, 1):
44         print(
45             '| {:>4} | {:<30} | {:<20} | {:>20} |'.format(
46                 idx,
47                 train.get('Пункт назначения ', ''),
48                 train.get('Номер поезда: ', ''),
49                 train.get('Время отправления: ', '0')
```

Рисунок 17.1 – Код программы индивидуального задания

18. Зафиксировал сделанные изменения в репозитории.

```
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git add .  
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git commit -m"add individual task"  
git: 'commit' is not a git command. See 'git --help'.  
  
The most similar command is  
    commit  
  
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->git commit -m"add individual task"  
[develop ce8e837] add individual task  
2 files changed, 79 insertions(+), 1 deletion(-)  
create mode 100644 pyCharm/individual.py  
C:\Users\tyt\Desktop\SE\laba11\laba-2.8-11->
```

Рисунок 18.1 – Коммит файлов в репозитории git

Контрольные вопросы

1. Каково назначение функций в языке программирования Python?

Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

2. Каково назначение операторов def и return?

В языке программирования Python функции определяются с помощью оператора def. Выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая

функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`.

5. Какие существуют способы передачи значений в функцию?

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым.

6. Как задать значение аргументов функции по умолчанию?

Для этого достаточно поставить знак равенства после имени параметра и указать его значение по умолчанию.

7. Каково назначение `lambda`-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые `lambda`-функции могут быть использованы везде, где требуется функция.

8. Как осуществляется документирование кода согласно PEP257?

Документирование кода в python - достаточно важный аспект, ведь от нее порой зависит читаемость и быстрота понимания вашего кода, как другими людьми, так и вами через полгода. PEP 257 описывает соглашения, связанные

со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис.

9. В чем особенность однострочных и многострочных форм строк документации?

Многострочные документации состоят из сводной строки (summary line), имеющей такую же структуру, как и однострочный docstring, после которой следует пустая линия, а затем более сложное описание