

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

«Управление потоками в Python»

Отчет по лабораторной работе № 2.23(10)

по дисциплине «Основы программной инженерии»

Выполнил студент группы ПИЖ-б-о-21-1

Халимендик Я. Д. « » 2023г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

Цель работы: приобретение навыков написания автоматизированных тестов на языке программирования Python версии 3.x.

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия IT 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.](#)

Owner * Yana-Kh / **Repository name *** Lab-2.23-OPJ

✔ Lab-2.23-OPJ is available.

Great repository names are short and memorable. Need inspiration? How about **musical-rotary-phone**?

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.](#)

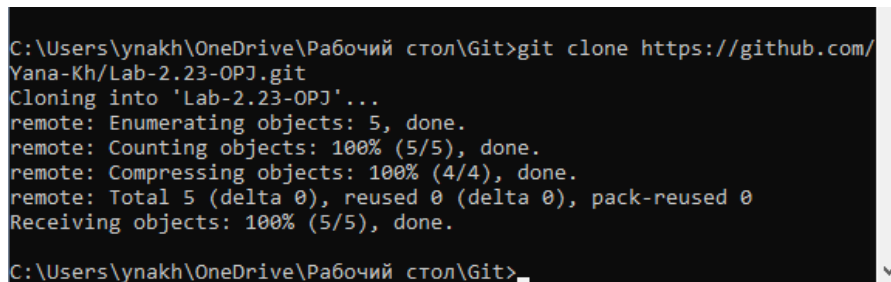
This will set **main** as the default branch. Change the default name in your [settings](#).

Info You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание репозитория

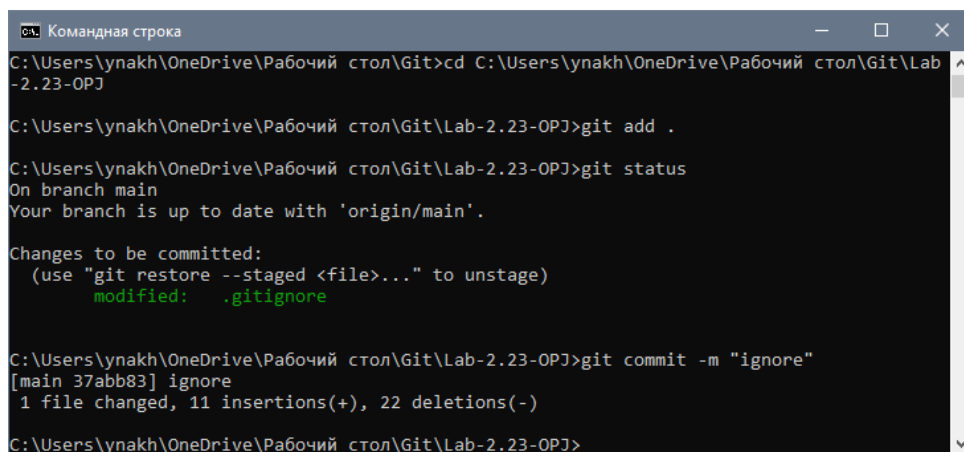
3. Выполните клонирование созданного репозитория.



```
C:\Users\ynakh\OneDrive\Рабочий стол\Git>git clone https://github.com/Yana-Kh/Lab-2.23-OPJ.git
Cloning into 'Lab-2.23-OPJ'...
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\ynakh\OneDrive\Рабочий стол\Git>
```

Рисунок 2 – Клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.



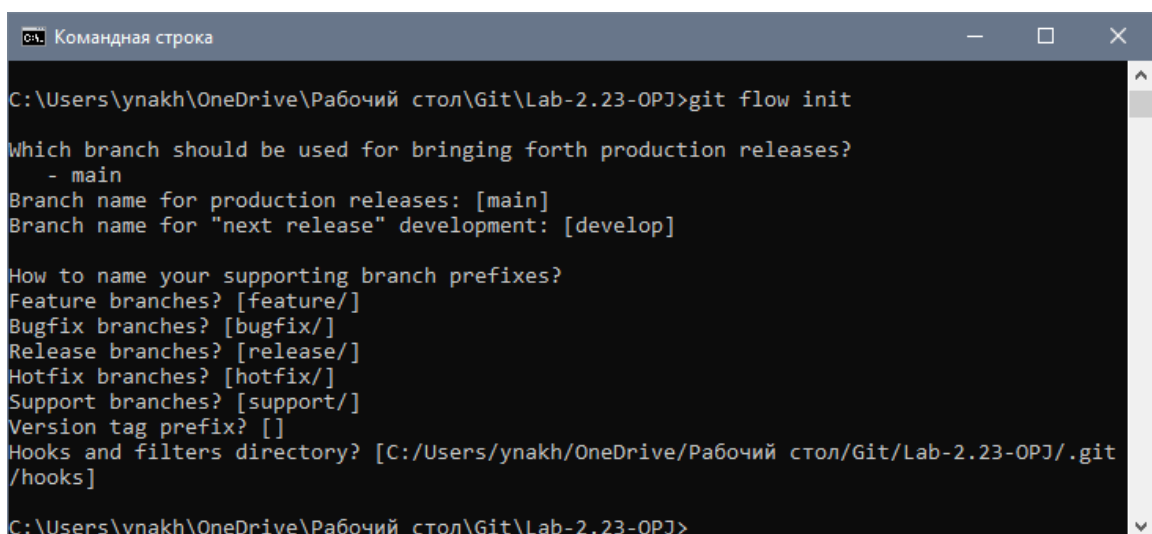
```
Командная строка
C:\Users\ynakh\OneDrive\Рабочий стол\Git>cd C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>git add .
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore

C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>git commit -m "ignore"
[main 37abb83] ignore
 1 file changed, 11 insertions(+), 22 deletions(-)
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>
```

Рисунок 3 – Дополнение файла .gitignore

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



```
Командная строка
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ynakh/OneDrive/Рабочий стол/Git/Lab-2.23-OPJ/.git/hooks]
C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-2.23-OPJ>
```

Рисунок 4 – Организация репозитория в соответствии с моделью git-flow

6. Создайте проект PyCharm в папке репозитория.

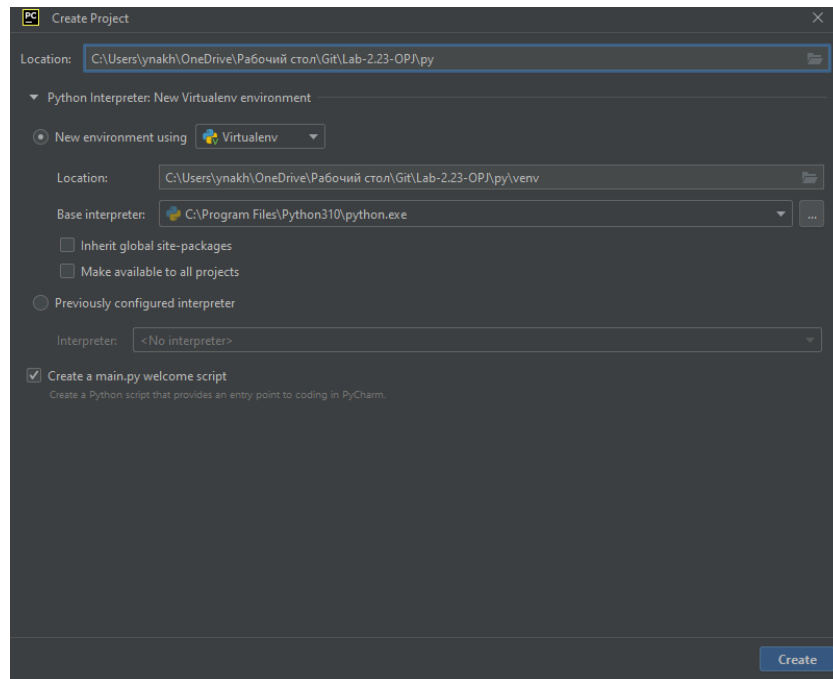


Рисунок 5 – Создание проекта PyCharm

7. Проработать примеры лабораторной работы. Создайте для них отдельные модули языка. Зафиксируйте изменения в репозитории.

Пример 1.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

if __name__ == '__main__':
    th = Thread(target=func)
    th.start()

    for i in range(5):
        print(f"from main thread: {i}")
        sleep(1)
```

```
from main thread: 1
from main thread: 2
from main thread: 3
from main thread: 4
from child thread: 1
from child thread: 2
from child thread: 3
from child thread: 4

Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

Пример 2.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

if __name__ == '__main__':
    th = Thread(target=func)
    print(f"thread status: {th.is_alive()}")
    th.start()
    print(f"thread status: {th.is_alive()}")
    sleep(5)
    print(f"thread status: {th.is_alive()}")
```

```
thread status: False
from child thread: 0thread status: True

from child thread: 1
from child thread: 2
from child thread: 3
from child thread: 4
thread status: False

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

Пример 3.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread
from time import sleep

class CustomThread(Thread):
    def __init__(self, limit):
        Thread.__init__(self)
        self.limit_ = limit

    def run(self):
        for i in range(self.limit_):
            print(f"from CustomThread: {i}")
            sleep(0.5)

if __name__ == '__main__':
    cth = CustomThread(3)
    cth.start()
```

```
from CustomThread: 0
from CustomThread: 1
from CustomThread: 2

Process finished with exit code 0
```

Рисунок 8 – Результат работы программы

Пример 4.

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread, Lock
from time import sleep

lock = Lock()
stop_thread = False

def infinit_worker():
    print("Start infinit_worker()")
    while True:
        print("--> thread work")
        lock.acquire()
        if stop_thread is True:
            break
        lock.release()
        sleep(0.1)
    print("Stop infinit_worker()")
```

```

if __name__ == '__main__':
    # Create and start thread
    th = Thread(target=infinite_worker)
    th.start()
    sleep(2)
    # Stop thread
    lock.acquire()
    stop_thread = True
    lock.release()

```

```

--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
Stop infinite_worker()

Process finished with exit code 0

```

Рисунок 9 – Результат работы программы

Пример 5.

Код:

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread
from time import sleep

def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)

if __name__ == '__main__':
    th = Thread(target=func, daemon=True)
    th.start()
    print("\nApp stop")

```

```
from child thread: 0
App stop

Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

8. Выполните индивидуальное задание. Приведите в отчете скриншоты работы программы решения индивидуального задания.

Задание: С использованием многопоточности для заданного значения найти сумму ряда S с точностью члена ряда по абсолютному значению

$\varepsilon = 10^{-7}$ и произвести сравнение полученной суммы с контрольным значением функции для двух бесконечных рядов. Номера вариантов необходимо уточнить у преподавателя:

Вариант 29(4):

$$S = \sum_{n=0}^{\infty} \left(\frac{1}{2^n} + \frac{1}{3^n} \right) x^{n-1} = \frac{5}{6} + \frac{13}{36}x + \frac{35}{216}x^2 + \dots; \quad x = -0,8; \quad y = \frac{5 - 2x}{6 - 5x + x^2}.$$

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Thread

def check_sum(x):
    y = (5-2*x) / (6 - 5*x + x**2)
    print(f"The result check-function is: {y}")

def s_sum(x):
    n = 1
    term = (1 / (2 ** n) + 1 / (3 ** n)) * (x ** (n - 1))
    epsilon = 1e-7
    ssum = term

    while abs(term) >= epsilon:
        n += 1
        term = (1 / (2 ** n) + 1 / (3 ** n)) * (x ** (n - 1))
        ssum += term

    print(f"The sum S is: {ssum}")

def main():
    x = -0.8
```



```

thread1 = Thread(target=check_sum(x))
thread1.start()
thread2 = Thread(target=s_sum(x))
thread2.start()

if __name__ == '__main__':
    main()

```

```

The result check-function is: 0.6203007518796991
The sum S is: 0.6203007273247924

Process finished with exit code 0

```

Рисунок 11 – Результат работы программы

9. Зафиксируйте сделанные изменения в репозитории.



Рисунок 10 – Фиксирование изменений в репозитории

Вопросы для защиты работы:

1. Что такое синхронность и асинхронность?

Синхронное выполнение программы подразумевает последовательное выполнение операций. Асинхронное – предполагает возможность независимого выполнения задач.

2. Что такое параллелизм и конкурентность?

Параллельность предполагает параллельное выполнение задач разными исполнителями, например: один человек занимается готовкой, другой приборкой.

Конкурентность предполагает выполнение нескольких задач одним исполнителем. Из примера с готовкой: один человек варит картошку и прибирается, при этом, в процессе, он может переключаться: немного

прибрався, пошел помешал-посмотрел на картошку, и делает он это до тех пор, пока все не будет готово.

3. Что такое GIL? Какое ограничение накладывает GIL?

GIL — это аббревиатура от Global Interpreter Lock – глобальная блокировка интерпретатора. Он является элементом эталонной реализации языка Python, которая носит название CPython. Суть GIL заключается в том, что выполнять байт код может только один поток. Это нужно для того, чтобы упростить работу с памятью (на уровне интерпретатора) и сделать комфортной разработку модулей на языке C.

Пока выполняется одна задача, остальные простаивают (из-за GIL), переключение происходит через определенные промежутки времени. Таким образом, в каждый конкретный момент времени, будет выполняться только один поток, несмотря на то, что у вас может быть многоядерный процессор (или многопроцессорный сервер), плюс ко всему, будет тратиться время на переключение между задачами.

4. Каково назначение класса Thread?

Он отвечает за создание, управление и мониторинг потоков.

5. Как реализовать в одном потоке ожидание завершения другого потока?

Если необходимо дождаться завершения работы потока(ов) перед тем как начать выполнять какую-то другую работу, то воспользуйтесь методом `join()`.

6. Как проверить факт выполнения потоком некоторой работы?

Для того, чтобы определить выполняет ли поток какую-то работу или завершился используется метод `is_alive()`.

7. Как реализовать приостановку выполнения потока на некоторый промежуток времени?

Для приостановки выполнения потока на некоторый промежуток времени в языке Python вы можете использовать функцию `time.sleep()`. Эта функция приостанавливает выполнение потока на указанное количество секунд.

8. Как реализовать принудительное завершение потока?

В Python у объектов класса `Thread` нет методов для принудительного завершения работы потока. Один из вариантов решения этой задачи – это создать специальный флаг, через который потоку будет передаваться сигнал остановки. Доступ к такому флагу должен управляться объектом синхронизации.

9. Что такое потоки-демоны? Как создать поток-демон?

Есть такая разновидность потоков, которые называются демоны (терминология взята из мира Unix-подобных систем). Python-приложение не будет закрыто до тех пор, пока в нем работает хотя бы один недемонический поток.