МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

***Циклова комісія програмування***

**Робочий план**

з курсу «Основи алгоритмізації та програмування»

Бєлан Нікіта Володимирович

*ПІБ студента*

студента групи **2П-20**

варіант \_\_4\_\_\_

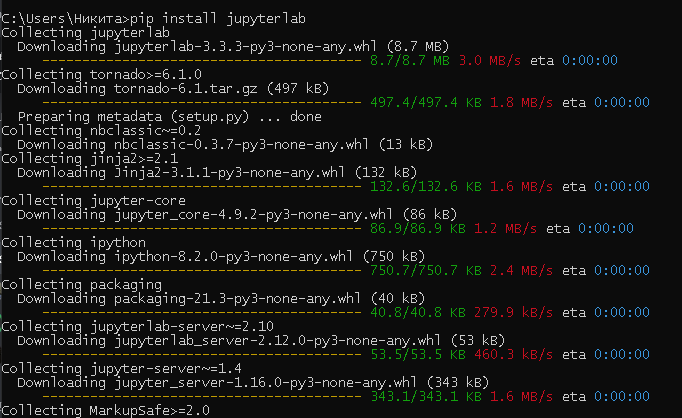
Викладач Марченко С. В.

|  |  |  |
| --- | --- | --- |
| **Вид роботи** | **Дата** | **Оцінка** |
| *Пром. контроль* |  |  |
| *Залік* |  |  |

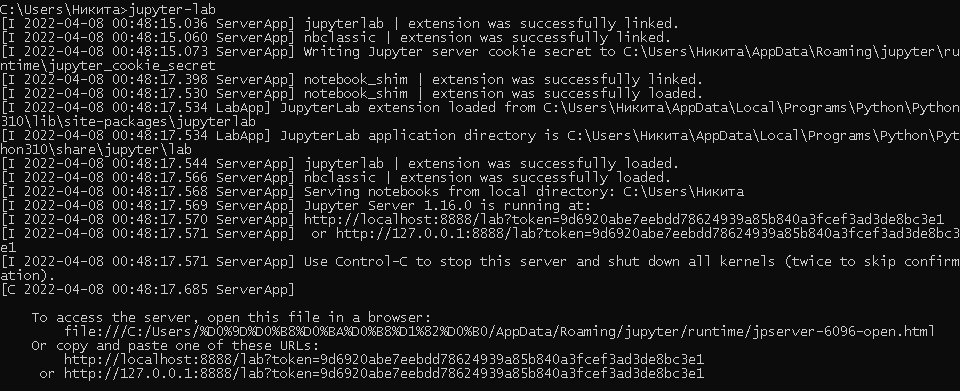
**Черкаси-2022**

**Завдання 1:**

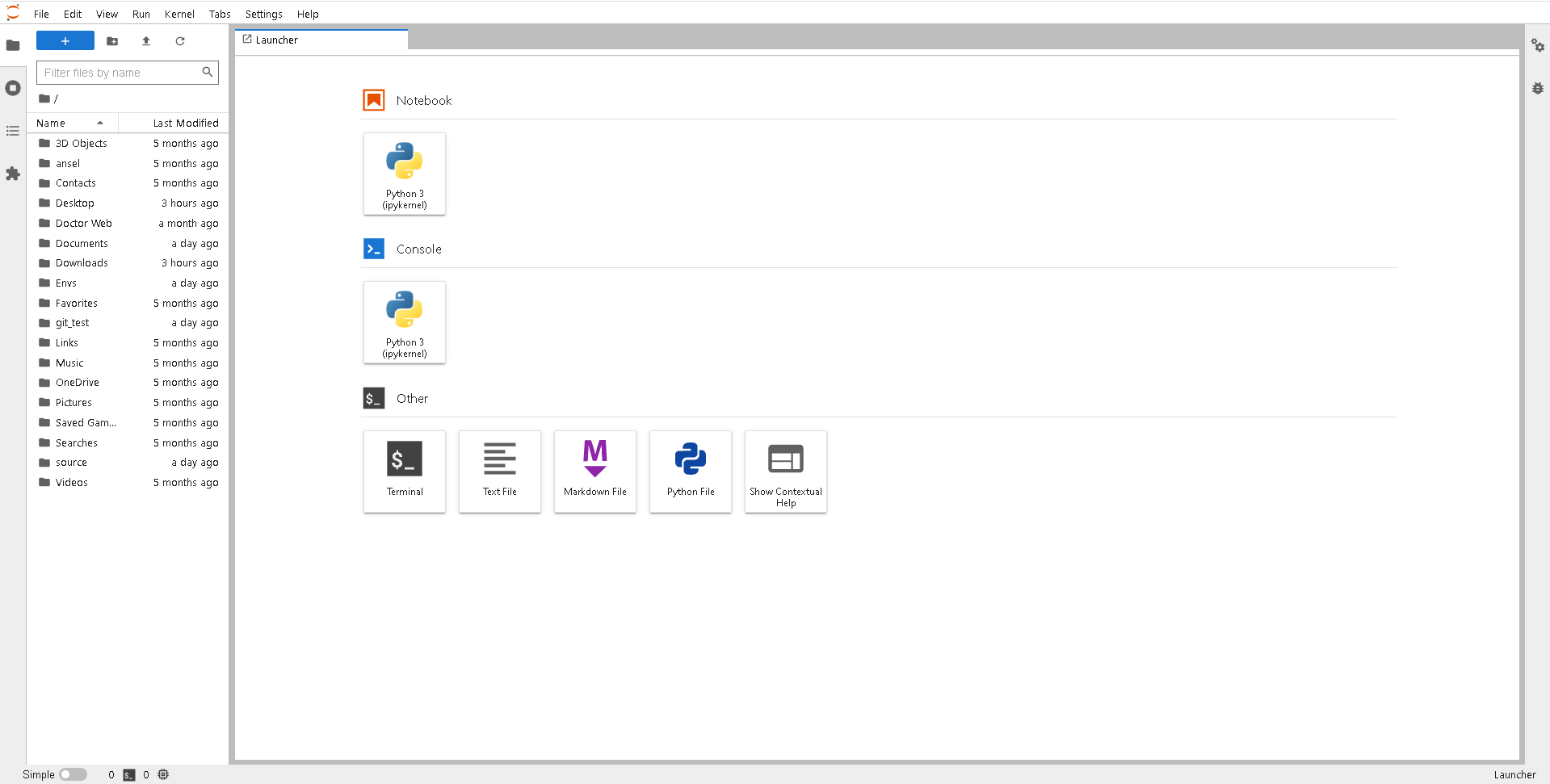
***0,6 бала* *(Jupyter Notebook)*** Ознайомтесь з можливостями роботи з інструментом Jupyter Notebook. Дослідіть питання практичної потреби та використання даного інструменту, опишіть у звіті переваги від його використання, а також продемонструйте інсталяцію та інтерактивну роботу з блокнотом Jupyter (файл .ipynb) на власній машині. За основу можете взяти [туторіал](https://www.dataquest.io/blog/jupyter-notebook-tutorial/).

****

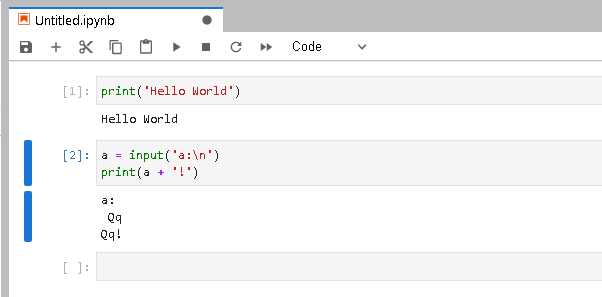
Інсталяція jypyterlab

****

Запуск jupyterlab

****

Вигляд початкової сторінки в jypyterlab

****

**Python у jypeterlab**

JupyterLab – гнучке, інтегроване середовище, що легко розширюється, підтримує одночасну роботу з кількома блокнотами Jupyter, текстовими файлами, датасетами, терміналами та іншими компонентами. Аналогічно класичним IDE у JupyterLab можна впорядковувати документи у робочій області у зручному порядку за допомогою вкладок та роздільників. JupyterLab підтримує відображення та редагування безлічі форматів даних: зображень, CSV, JSON, Markdown, PDF, Vega, Vega-Lite і так далі. Для швидкої навігації за документами в JupyterLab є гарячі клавіші, що настроюються, а також можливість використання стандартних поєднань з vim, emacs і Sublime Text. JupyterLab може бути доповнений за допомогою розширень, що дозволяють налаштувати різні частини JupyterLab, включаючи теми, редактори файлів і додаткові компоненти.

**Запуск**

Запуск сервера JupyterLab з терміналу здійснюється наступною простою командою:

****

**Завдання 2:**

*1 бал* ***(Комплексні основи ООП)*** Створіть новий клас Student, породжений класом Person. Визначте атрибути, які має лише студент, і не мають інші люди. Наприклад, студент відвідує навчальний заклад, має рік вступу, перелік поточних/пройдених курсів з оцінками, середній бал за кожну сесію (GPA) та ін.

Створіть об’єкт класу Student та покажіть, що наслідування виконується коректно. Задайте деяким атрибутам для студента значення, записані лише в класі Person або лише в класі Student. Виведіть значення цих атрибутів.

Доповніть архітектуру додатку класами, які описують кафедру навчання студента. Кожна кафедра має назву та перелік викладачів, які на ній працюють. Кожен з викладачів має свій набір курсів для викладання. Загалом кафедра пропонує набір курсів (потрібен клас для відповідного опису), які студент має завершити протягом свого навчання. Кожен курс мінімально має свій код, назву, кількість навчальних кредитів (від 1 до 6), прикріплених до нього викладачів (достатньо ПІБ), час початку та час завершення.

Збережіть класи Person та Student в окремому файлі (person.py), а код, що використовує ці класи, в чотирьох окремих файлах:

1. Тут використовуйте синтаксис «from module\_name import ClassName», щоб запустити Вашу програму.
2. Тут використовуйте синтаксис «import module\_name».
3. Використовуйте синтаксис «import module\_name as different\_local\_module\_name».
4. Використовуйте синтаксис «import \*».

**person.py:**

**class Person:**

**def \_\_init\_\_(self, first\_name, second\_name, age):**

**self.first\_name = first\_name**

**self.second\_name = second\_name**

**self.age = age**

**def \_\_str\_\_(self):**

**return f'Привіт, мене звати {self.first\_name} {self.second\_name}'**

**class Teacher(Person):**

**def \_\_init\_\_(self, first\_name, second\_name, age):**

**super().\_\_init\_\_(first\_name, second\_name, age)**

**self.courses = []**

**class Student(Person):**

**def \_\_init\_\_(self, first\_name, second\_name, age, school, yearEntry, gpa, department):**

**super().\_\_init\_\_(first\_name, second\_name, age)**

**self.school = school**

**self.yearEntry = yearEntry**

**self.gpa = gpa**

**self.departament = department**

**def \_\_str\_\_(self):**

**return f'first name: {self.first\_name}\nsecond name: {self.second\_name}\nage:{self.age}\nschool:{self.school}\nyearEntry:{self.yearEntry}\nGPA: {self.gpa}'**

**main.py:**

**#import person**

**#from person import Student**

**#import person as p**

**from person import \***

**class Course:**

**def \_\_init\_\_(self):**

**self.id = ''**

**self.title = ''**

**self.credit = 0**

**self.teachers = []**

**class Department:**

**def \_\_init\_\_(self):**

**self.title = ''**

**self.teachers = []**

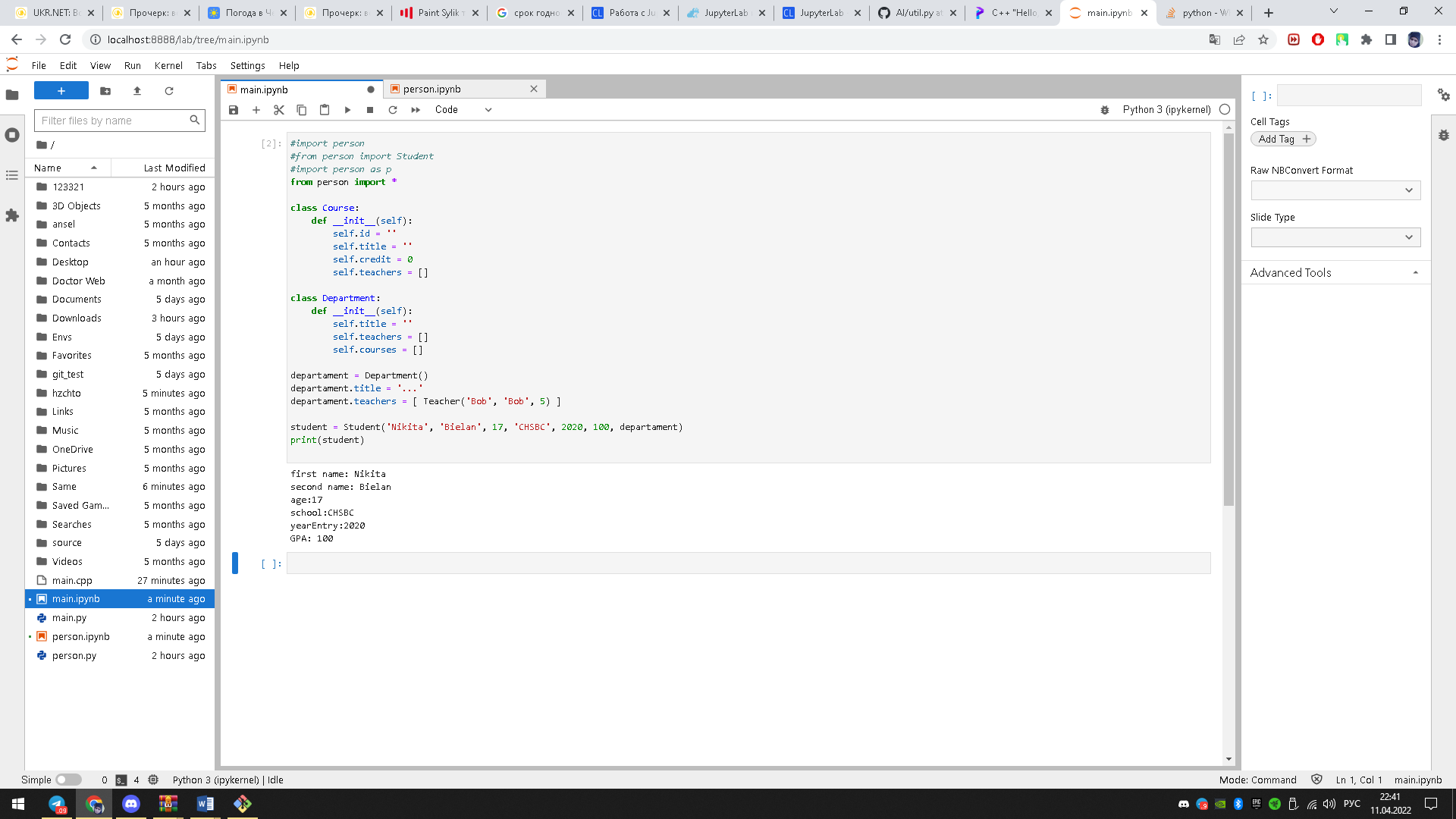
**self.courses = []**

**departament = Department()**

**departament.title = '...'**

**departament.teachers = [ Teacher('Bob', 'Bob', 5) ]**

**student = Student('Nikita', 'Bielan', 17, 'CHSBC', 2020, 100, departament)**

print(student)

**Завданяя 3:**

*0,7 бала* Напишіть програму, яка створюватиме стек та дозволятиме виконувати операції з ним. На її базі реалізуйте можливість перевірки синтаксису програм мовою С: встановлення правильності послідовності фігурних дужок ({…}). Програма повинна отримувати текст програми мовою С, при знаходженні символу “{“ додавати його в стек, а при появі “}” – виконувати зняття елементу зі стеку. Перевірка стеку на пустоту встановить, чи правильно вжиті фігурні дужки в програмі.

**import os**

**text = ''**

**try:**

**f = open('main.cpp', 'r')**

**text = f.read()**

**f.close()**

**print(text)**

**except Exception as e:**

**print('Не удалось прочитать файл:', e)**

**os.abort()**

**class Stack:**

**stack = []**

**top = 0**

**def push(self, el):**

**self.top += 1**

**def pop(self):**

**if self.top <= 0:**

**raise "ERROR"**

**self.top -= 1**

**def empty(self):**

**return self.top == 0**

**stack = Stack()**

**for ch in text:**

**if ch == '{':**

**stack.push(ch)**

**elif ch == '}':**

**stack.pop()**

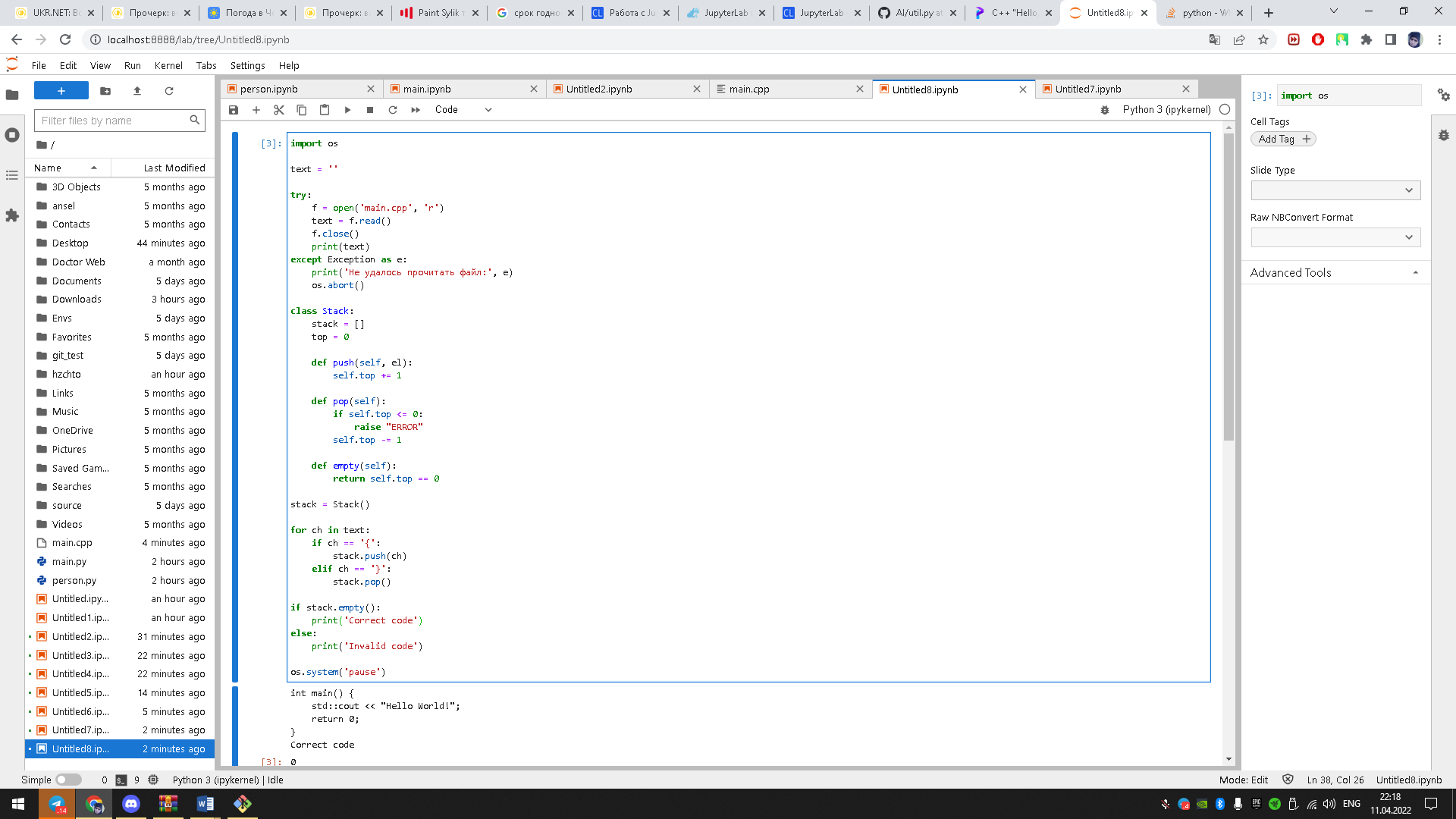
**if stack.empty():**

**print('Сorrect code')**

**else:**

**print('Invalid code')**

**os.system('pause')**



**Завдання 4:**

*0,7 бала* Напишіть клас, який описуватиме створення черги та дозволятиме виконувати операції з нею. Створіть чергу з чисел та продемонструйте роботу даних операцій.

**class Queue:**

**queue = [None]\*100**

**tail = 0**

**head = 0**

**def enqueue(self, el):**

**self.queue[self.tail] = el**

**self.tail += 1**

**def dequeue(self):**

**x = self.queue[self.head]**

**self.head += 1**

**return x**

**q = Queue()**

**q.enqueue('1')**

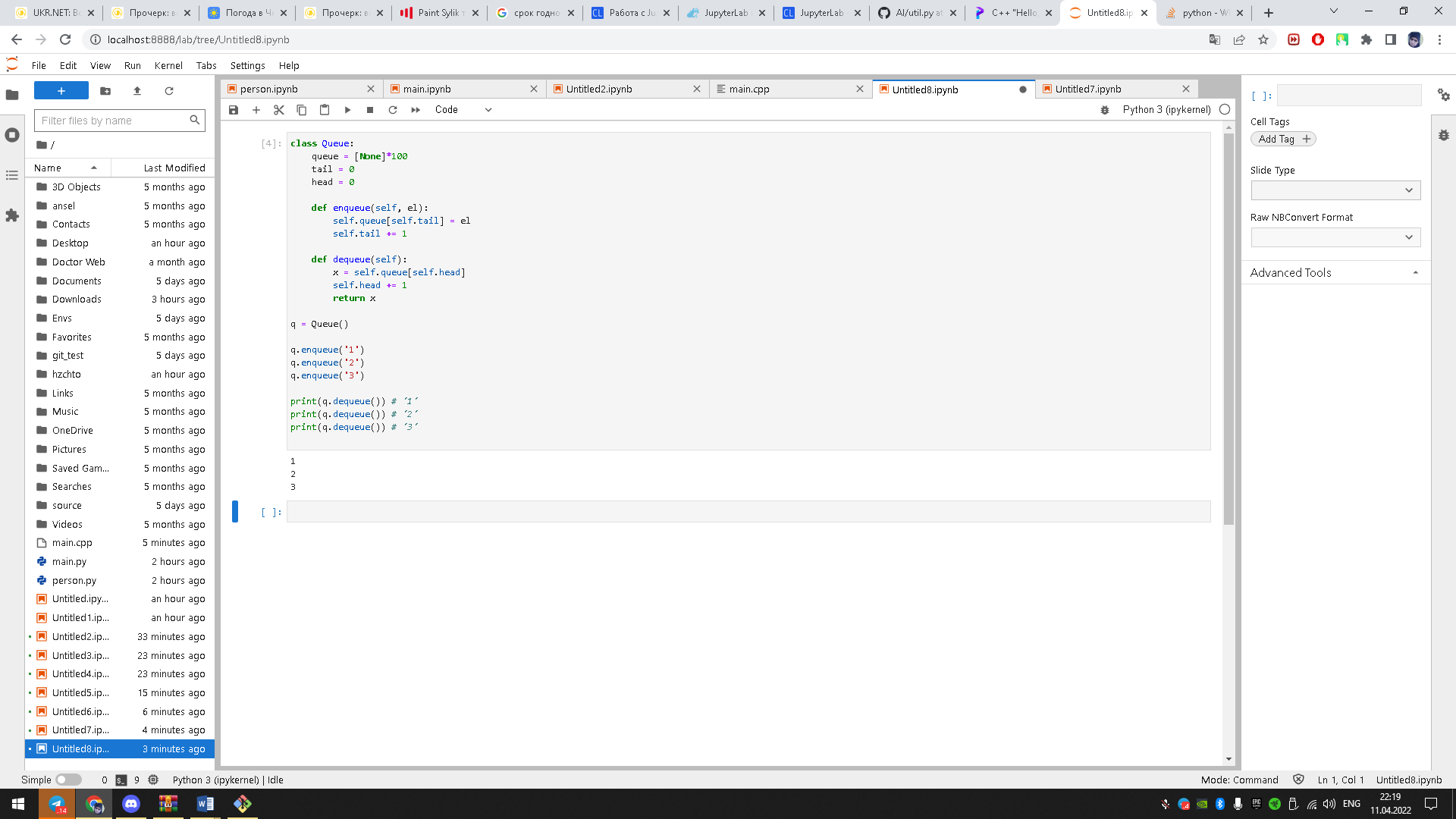
**q.enqueue('2')**

**q.enqueue('3')**

**print(q.dequeue()) # '1'**

**print(q.dequeue()) # '2'**

**print(q.dequeue()) # '3'**



.