Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 5

з навчальної дисципліни

“Cкриптові мови програмування (Python)”

Об’єктно-орієнтоване програмування у Python

ЗАВДАННЯ ВИДАВ Лектор Мелешко Є. В.

ВИКОНАВ

студент академічної групи КБ-20

Прокопенко Є.С.

ПЕРЕВІРИВ

Асистент Савеленко О. К.

Кропивницький – 2021

**Мета:**набути навичок роботи з класами у Python

**Варіант – 16**

**Завдання до лабораторної роботи:**

1. Розробити клас "домашня бібліотека". Реалізувати можливість роботи з довільним числом книг, пошуку по книгах за декількома параметрами (за автором, за роком видання, за жанром тощо), додавання книг у бібліотеку, видалення книг з неї, доступу до книги за номером. Написати програму, що буде демонструвати всі розроблені елементи класу.

2. Розробити клас для представлення відомостей про успішність студента. Об'єкт класу має містити поля для збереження імені студента та балів, отриманих ним за виконання лабораторних робіт та індивідуального творчого завдання.

Забезпечити наступні методи класу:

* конструктор, який приймає рядок ім'я\_студента та словник, що містить налаштування курсу у наступному форматі:

1) максимально можлива кількість балів за здачу індивідуального творчого завдання;

2) максимально можлива кількість балів за здачу однієї лабораторної роботи;

3) кількість лабораторних робіт в курсі;

4) частка балів від максимуму, яку необхідно набрати для отримання екзамену автоматом.

* метод, за допомогою якого вносяться дані про кількість спроб здати лабораторну роботу та оцінка за останню спробу.
* метод, за допомогою якого вносяться дані про кількість спроб здати індивідуальне творче завдання та оцінка за останню спробу.
* метод, який повертає кортеж (tuple), що містить дійсне число (суму балів студента за проходження курсу), та логічне значення True або False в залежності від того, чи достатньо цих балів для отримання оцінки за екзамен автоматом.

3. Розробити клас, який наслідує функціональність стандартного типу str і містить 2 нових методи:

1) метод, який приймає 1 аргумент s та повертає True або False в залежності від того, чи містить рядок повтори послідовностей символів довжиною від 3 символів.

2) метод, який повертає True або False в залежності від того, чи є рядок паліндромом. Регістрами символів нехтувати. Порожній рядок вважати паліндромом.

4. Розробити клас "колода карт", що буде включати закритий масив елементів класу "карта". В карті буде зберігатися масть та номер. При створенні екземпляру класу "колода карт", карти у колоді розташовуються випадковим чином. Забезпечити можливість виведення карти за номером розташування у колоді, виведення всіх карт, перемішування колоди, видачі однієї карти з колоди, видачі 6 карт з колоди. Написати програму, що буде демонструвати всі розроблені елементи класу.

5. Розробити клас "англо-російський словник", забезпечити можливість зберігання декількох варіантів перекладу для кожного слова. Забезпечити можливість виведення всіх варіантів перекладу введеного англійського слова.

6. Створити абстрактний клас "Транспортний засіб". На його основі реалізувати класи "Літак", "Автомобіль" та "Корабель". Класи повинні мати можливість задавати та отримувати координати і параметри засобів пересування (вартість, швидкість, рік випуску тощо) задати за допомогою полів. Для літака повинна бути визначена висота, для літака та корабля – кількість пасажирів, для корабля – порт приписки. Динамічні характеристики задати за допомогою методів.і

**Завдання 1**

class Book:

def \_\_init\_\_(self, name = " ", year = 0, genre = " "):

if name != " " and 0<year<=2021 and genre != " ":

self.name = name

self.genre = genre

self.year = year

else:

print("Помилка додавання книги!")

class Lib:

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

self.library = []

def addBook(self, Book):

if Book not in self.library:

self.library.append(Book)

print("Книга " + str(Book.name) + " була успішно доданa до бібліотеки")

def findBook(self, info):

for search in self.library:

if search.name == info or search.year == info or search.genre == info:

print("Книга знайдена!")

print("Назва: "+str(search.name) + "\nРік: " + str(search.year) + "\nЖанр: " + str(search.genre))

break

else:

print("Книгу не вдаєтся знайти...")

def delBook(self, b):

for search in self.library:

if b is search:

self.library.remove(b)

print("Книгу успішно видалено.")

if \_\_name\_\_ == '\_\_main\_\_':

**library = Lib()**

**name = input("Введіть назву книги: ")**

**year = int(input("Введіть рік редакції книги: "))**

**genre = input("Введіть жанр книги: ")**

**book = Book(name, year, genre)**

**library.addBook(book)**

**library.findBook(name)**

**library.delBook(book)**

**library.findBook(name)**

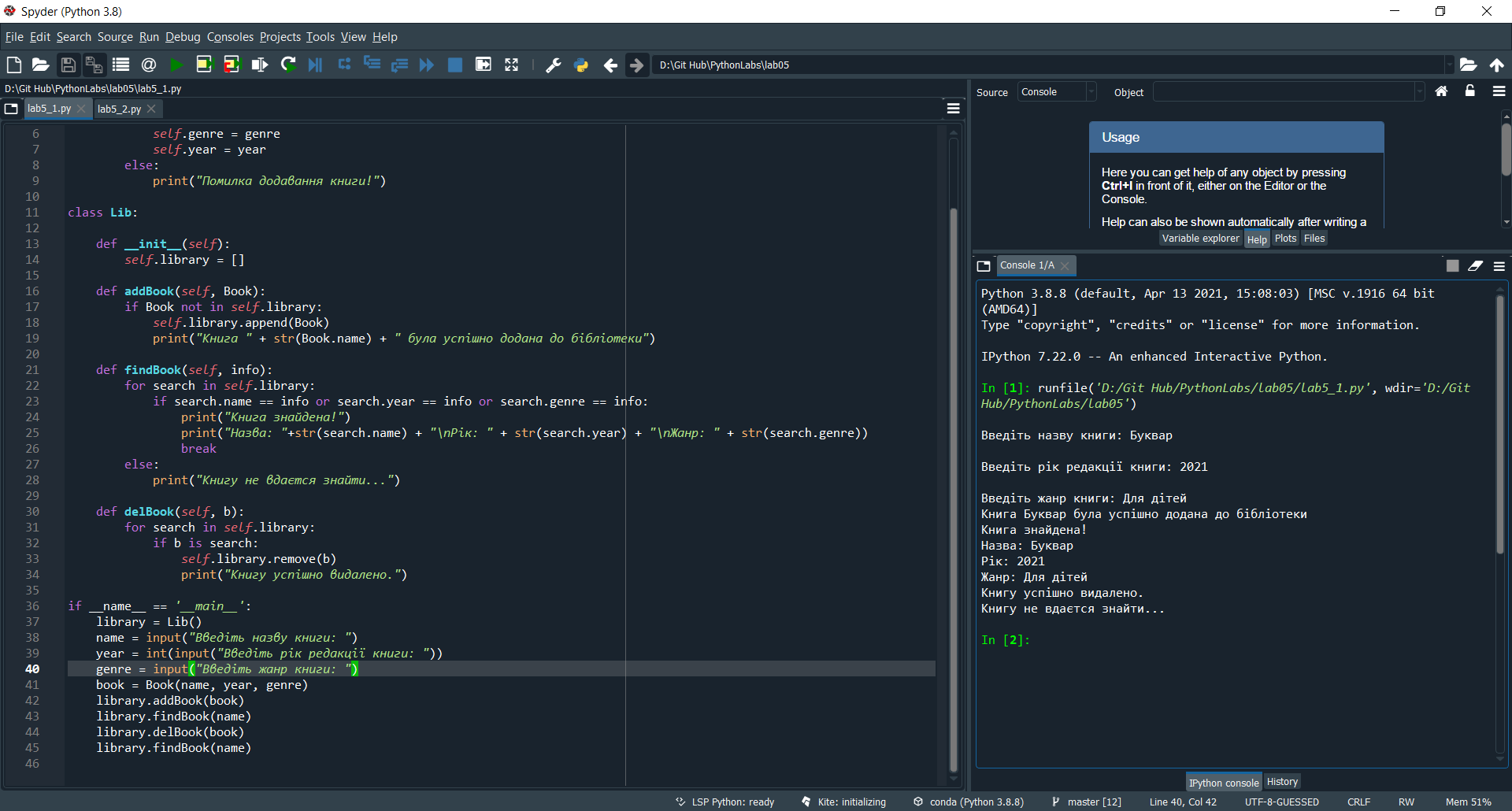


Рисунок 1 – Результат виконання завдання 1

**Завдання 2**

**Лістинг:**

import random

class Student:

def \_\_init\_\_(self, name, \*\*dictionary):

if isinstance(dictionary["max\_marks\_ind"], int) and isinstance(dictionary["max\_marks\_lab"], int):

if dictionary["count\_labs"] and dictionary["marks\_for\_passing"]:

self.name = name

self.max\_marks\_i = dictionary["max\_marks\_ind"]

self.max\_marks\_l = dictionary["max\_marks\_lab"]

self.count\_labs = dictionary["count\_labs"]

self.marks\_part = dictionary["marks\_for\_passing"]

self.marks\_for\_labs = []

self.mark\_for\_ind = 0

print("Студент " + name + " був успішно створений!")

else:

print("Помилка створення студента!")

else:

print("Помилка створення студента!")

def get\_tries\_lab(self, try\_pass, mark):

if int(mark) < int(self.max\_marks\_l) and self.max\_marks\_l != None:

for i in range(int(self.count\_labs)):

passed\_try = random.randint(int(0), int(try\_pass))

if passed\_try >= 1:

self.marks\_for\_labs.append(random.randint(int(mark), int(self.max\_marks\_l)))

else:

self.marks\_for\_labs.append(mark)

return 1

else:

return 0

def get\_tries\_ind(self, try\_pass, mark):

if int(mark) < int(self.max\_marks\_i) and self.max\_marks\_i != None:

passed\_try = random.randint(int(0), int(try\_pass))

if passed\_try >= 1:

self.mark\_for\_ind = random.randint(int(mark), int(self.max\_marks\_i))

else:

self.mark\_for\_ind = mark

return 1

else:

return 0

def get\_res(self):

num = int(self.mark\_for\_ind)

for mark in self.marks\_for\_labs:

num += int(mark)

if num >= self.marks\_part:

log\_val = True

else:

log\_val = False

return (num, log\_val)

def input\_int\_digit(string):

"""

Return int number

Parameters

----------

string : what write in input.

Returns

-------

number : int number.

"""

while True:

number = input(string)

if(number.isdigit() and int(number) >= 0):

return number

if \_\_name\_\_ == '\_\_main\_\_':

name = input("Введіть ім'я студента: ")

mark\_lab = input\_int\_digit("Введіть максимальну оцінку за здачу лабораторної роботи: ")

mark\_ind = input\_int\_digit("Введіть максимальну оцінку за здачу індивідуального завдання: ")

labs = input\_int\_digit("Введіть кількість лабораторних робіт: ")

res\_mark = input\_int\_digit("Введіть кількість балів, що потрібно набрати студенту: ")

dict = {'max\_marks\_ind':int(mark\_ind),'max\_marks\_lab':int(mark\_lab),'count\_labs':int(labs), 'marks\_for\_passing':int(res\_mark)}

Stud = Student(name, \*\*dict)

try\_pass\_l = input\_int\_digit("Введіть кількість спроб здати лабораторну роботу: ")

min\_mark\_l = input\_int\_digit("Введіть оцінку за здачу лабораторної роботи з останньої спроби: ")

try\_pass\_i = input\_int\_digit("Введіть кількість спроб здати індивідуальну роботу: ")

min\_mark\_i = input\_int\_digit("Введіть оцінку за здачу індивідуальної роботи з останньої спроби: ")

if Stud.get\_tries\_lab(try\_pass\_l, min\_mark\_l) and Stud.get\_tries\_ind(try\_pass\_i, min\_mark\_i):

print("Успішність студента:" + str(Stud.get\_res()))

else:

print("Помилка обробки данних студента :( ")

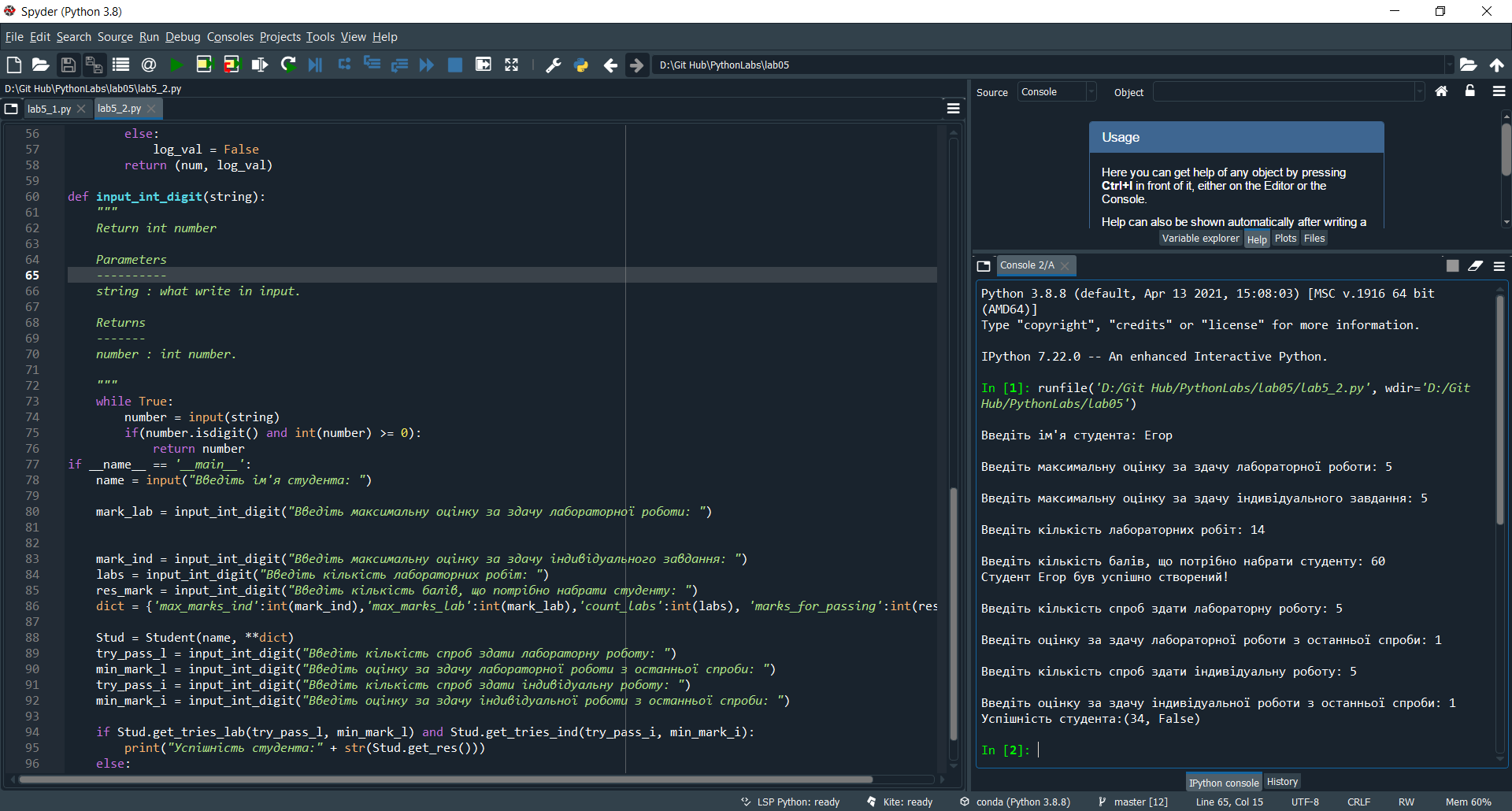


Рисунок 2 – Результат виконання завдання 2

**Завдання 3**

**Лістинг:**

class ModString (str):

def \_\_init\_\_(self, input):

self.s = input

def substr(self):

result = False

s = self.s.lower()

for i in range(len(s)-3):

sub = s[i:i+3]

if sub in s[3:len(s)-1]:

result = True

break

if result:

return True

else:

return False

def polindrom(self):

s = self.s.lower()

if s[::-1] == s:

return True

else:

return False

stroka = ModString("pythonpythonnohtypnohtyp")

print("Введена строка: " + str(stroka))

print("Чи введена строка містить у собі повтори від 3 символів: " + str(stroka.substr()))

print("Чи введена строка є поліндромом: " + str(stroka.polindrom()))

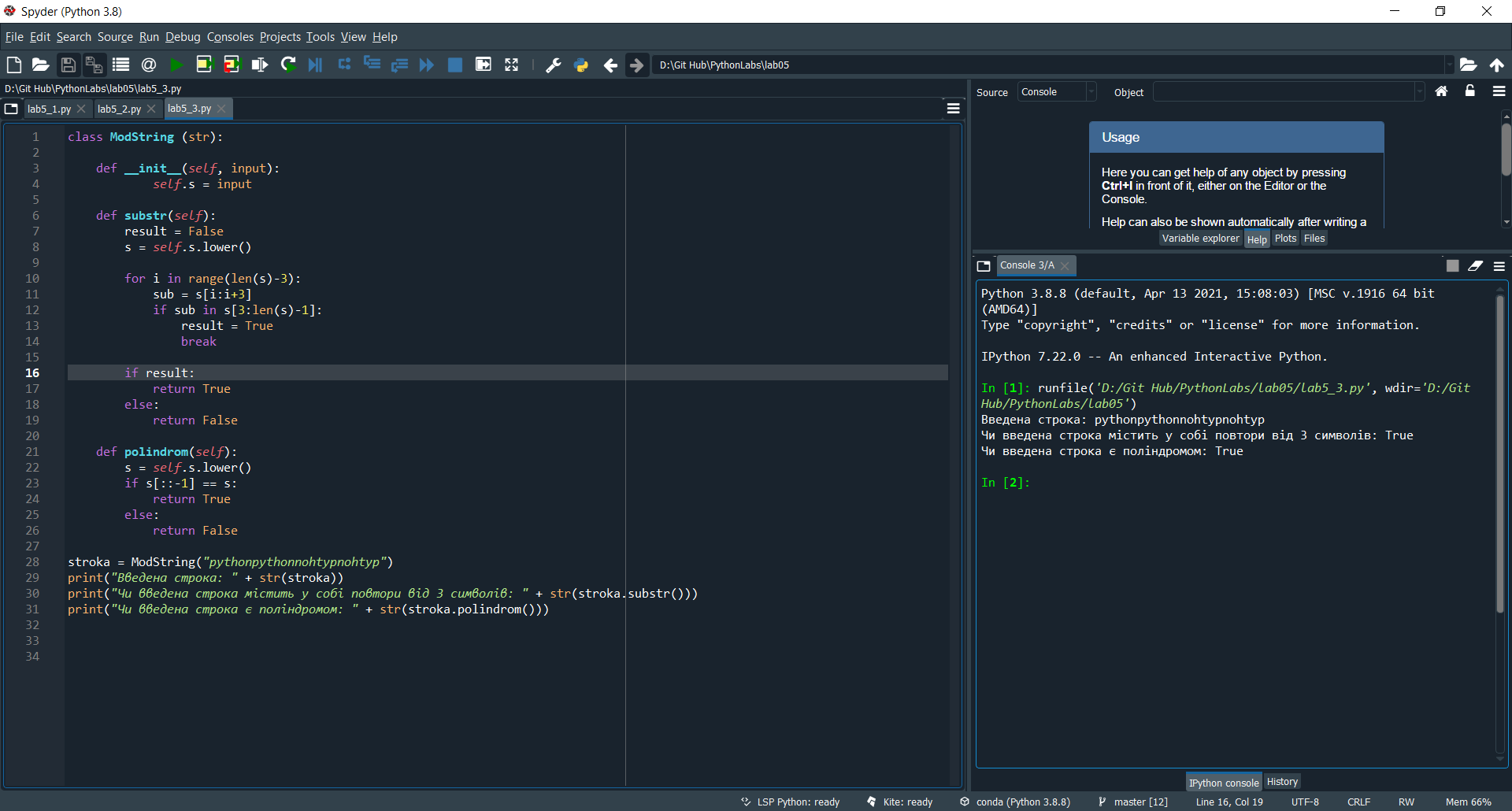


Рисунок 3 – Результат виконання завдання 3

**Завдання 4**

**Лістинг:**

import random

class Card:

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

self.mast = random.choice(["Чирва", "Піка", "Трефа", "Бубна"])

self.num = random.choice([6, 7, 8, 9, 10, "Валет", "Дама", "Kороль", "Туз"])

def show(self):

print("Масть: " + self.mast + "\n" + "Ціна: " + str(self.num))

class Stack:

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

self.stack = []

for i in range(35):

random\_card = Card()

if random\_card not in self.stack:

self.stack.append(random\_card)

def findByIndex(self, index):

if 0<=index<=35:

print(str(index) + " картка зі стопки : " )

self.stack[index].show()

else:

print("Індекс вийшов за межі діапазону")

def showStack(self):

for index in range(35):

self.stack[index].show()

def getSixCards(self):

for i in range(6):

random.choice(self.stack).show()

def transport(self):

new = random.sample(self.stack,len(self.stack))

for elem in new:

elem.show()

if \_\_name\_\_ == '\_\_main\_\_':

**stack = Stack()**

**stack.showStack()**

**print()**

**stack.findByIndex(12)**

**print("\nШість випадкових карт\n")**

**stack.getSixCards()**

**print("\n! Нова колода !\n")**

**stack.transport()**

**Результат тестового запуску:**

Масть: Чирва

Ціна: Kороль

Масть: Піка

Ціна: Kороль

Масть: Бубна

Ціна: 6

Масть: Трефа

Ціна: 10

Масть: Чирва

Ціна: 8

Масть: Чирва

Ціна: 9

Масть: Чирва

Ціна: Дама

Масть: Бубна

Ціна: Валет

Масть: Чирва

Ціна: 8

Масть: Трефа

Ціна: 6

Масть: Піка

Ціна: Kороль

Масть: Бубна

Ціна: 8

Масть: Чирва

Ціна: 6

Масть: Бубна

Ціна: Дама

Масть: Чирва

Ціна: Туз

Масть: Трефа

Ціна: Дама

Масть: Піка

Ціна: 7

Масть: Піка

Ціна: 6

Масть: Чирва

Ціна: Kороль

Масть: Піка

Ціна: Дама

Масть: Чирва

Ціна: 6

Масть: Трефа

Ціна: 10

Масть: Піка

Ціна: 8

Масть: Бубна

Ціна: 10

Масть: Піка

Ціна: 8

Масть: Чирва

Ціна: 7

Масть: Трефа

Ціна: 9

Масть: Чирва

Ціна: 8

Масть: Піка

Ціна: 10

Масть: Бубна

Ціна: Kороль

Масть: Піка

Ціна: 7

Масть: Трефа

Ціна: Дама

Масть: Бубна

Ціна: Валет

Масть: Піка

Ціна: Валет

Масть: Піка

Ціна: 10

12 картка зі стопки :

Масть: Чирва

Ціна: 6

Шість випадкових карт

Масть: Трефа

Ціна: 9

Масть: Чирва

Ціна: Дама

Масть: Чирва

Ціна: 7

Масть: Трефа

Ціна: Дама

Масть: Трефа

Ціна: 9

Масть: Трефа

Ціна: 6

! Нова колода !

Масть: Чирва

Ціна: 8

Масть: Піка

Ціна: Kороль

Масть: Бубна

Ціна: Валет

Масть: Чирва

Ціна: Kороль

Масть: Трефа

Ціна: Дама

Масть: Трефа

Ціна: 9

Масть: Чирва

Ціна: Дама

Масть: Чирва

Ціна: 6

Масть: Трефа

Ціна: 6

Масть: Чирва

Ціна: 7

Масть: Піка

Ціна: 7

Масть: Трефа

Ціна: Дама

Масть: Бубна

Ціна: Kороль

Масть: Піка

Ціна: 10

Масть: Піка

Ціна: 7

Масть: Трефа

Ціна: 10

Масть: Піка

Ціна: Kороль

Масть: Бубна

Ціна: Дама

Масть: Піка

Ціна: 8

Масть: Піка

Ціна: 6

Масть: Піка

Ціна: Валет

Масть: Піка

Ціна: Дама

Масть: Бубна

Ціна: 10

Масть: Чирва

Ціна: Kороль

Масть: Чирва

Ціна: 8

Масть: Піка

Ціна: 10

Масть: Чирва

Ціна: 6

Масть: Бубна

Ціна: 8

Масть: Піка

Ціна: 8

Масть: Чирва

Ціна: 8

Масть: Бубна

Ціна: 6

Масть: Трефа

Ціна: 10

Масть: Чирва

Ціна: 9

Масть: Бубна

Ціна: Валет

Масть: Чирва

Ціна: Туз

**Завдання 5**

class Dictionary:

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

self.dictionary = dict()

def addWordTranslate(self, word, translation):

if word != "" or translation != "":

if word not in self.dictionary.keys():

self.dictionary[word] = translation

print("Слово \"" + word + "\" було успішно додано до словника")

else:

print("Слово вже існує у словнику!")

else:

print("Помилка додавання слова до словника!")

def showTranslation(self, word):

if word in self.dictionary.keys():

print("Переклади для слова \"" + word + '\":')

print(self.dictionary[word])

else:

print("Помилка, слова \"" + word + "\" немає у словнику!")

if \_\_name\_\_ == '\_\_main\_\_':

**Dict = Dictionary()**

**Dict.addWordTranslate("doll", ["гарненька дитина", "дівчина", "коханка", "лялька"])**

**Dict.addWordTranslate("doll", "дол")**

**Dict.addWordTranslate("brother", "Брат")**

**Dict.showTranslation("brother")**

**Dict.showTranslation("doll")**

**Dict.showTranslation("Egor")**

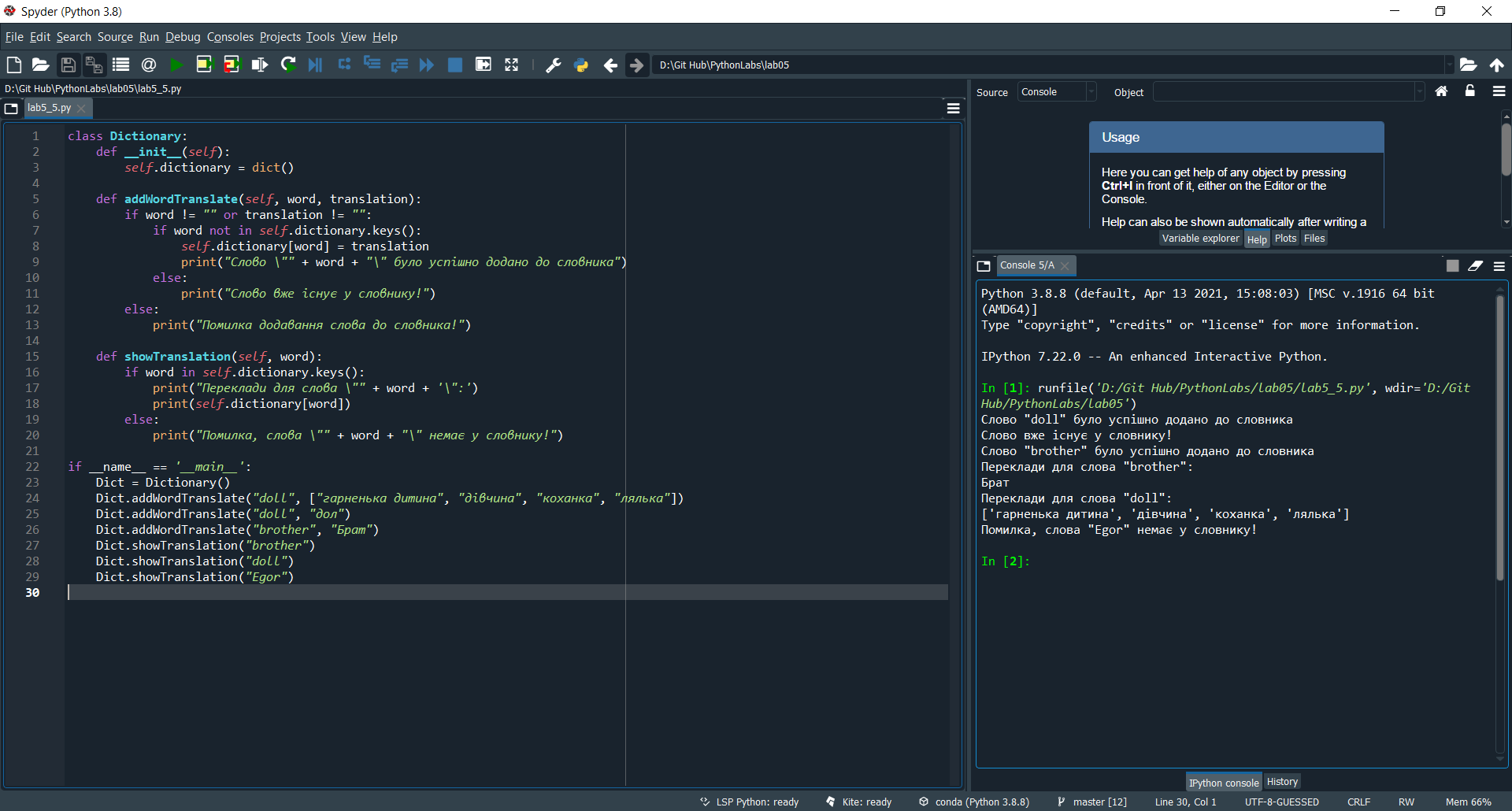


Рисунок 4 – Результат виконання завдання 5

**Завдання 6**

**Лістинг:**

class Transport:

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

self.value = 0

self.speed = 0

self.year = 0

self.coord = ""

def setVal(self, input):

self.value = input

def setSpeed(self, input):

self.speed = input

def setYear(self, input):

self.year = input

def setCoord(self, input):

self.coord = input

class Car (Transport):

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

self.value = 0

self.speed = 0

self.year = 0

self.coord = ""

def getCarCharacteristics(self):

print("\nХарактеристики автомобіля\n"

+ "Ціна: " + str(self.value)

+ "\nШвидкість: " + str(self.speed)

+ "\nРік: " + str(self.year)

+ "\nКоординати: " + self.coord)

class Plane (Transport):

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

self.value = 0

self.speed = 0

self.passengers = 0

self.coord = ""

self.year = 0

self.height = 0

def setHeight(self, input):

self.height = input

def setPassengers(self, input):

self.passengers = input

def getPlaneCharacteristics(self):

print("\nХарактеристика Літака\n"

+ "Ціна: " + str(self.value)

+ "\nШвидкість: " + str(self.speed)

+ "\nРік: " + str(self.year)

+ "\nКоординати: " + self.coord

+ "\nВисота: " + str(self.height)

+ "\nКількість пасажирів: " + str(self.passengers))

class Ship (Plane):

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

self.value = 0

self.speed = 0

self.year = 0

self.port = 0

self.coord = ""

self.passangers = 0

def setPort(self, input):

if isinstance(input, str):

self.port = input

def getShipCharacteristics(self):

print("\nХарактеристика корабля\n"

+ "Ціна: " + str(self.value)

+ "\nШвидкість: " + str(self.speed)

+ "\nРік: " + str(self.year)

+ "\nКоординати: " + self.coord

+ "\nПорт: " + self.port

+ "\nКількість пасажирів: " + str(self.passengers))

if \_\_name\_\_ == '\_\_main\_\_':

**ship = Ship()**

**ship.setVal(250000)**

**ship.setSpeed(200)**

**ship.setYear(1956)**

**ship.setCoord("18:30:245")**

**ship.setPassengers(56)**

**ship.setPort("Одеський морський порт")**

**ship.getShipCharacteristics()**

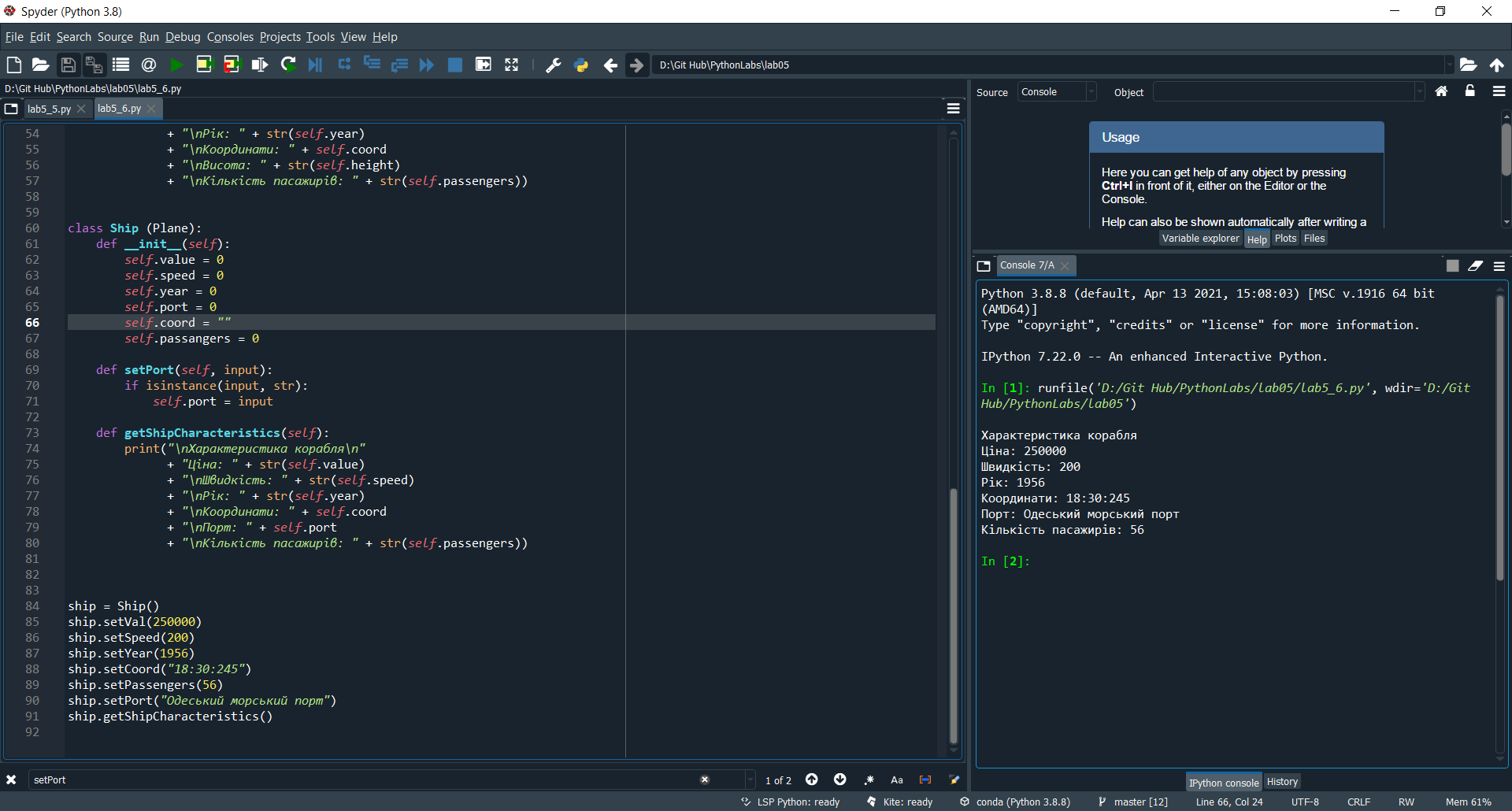


Рисунок 5 – Результат виконання завдання 6