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

Микитенко Д. Ю.

ПЕРЕВІРИВ

Асистент Савеленко О. К.

Кропивницький – 2021

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

**Варіант – 12**

**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("Error of adding book")

class Lib:

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

self.array = list()

def AddBook(self, Book):

if Book not in self.array:

self.array.append(Book)

print("Book" + str(Book.name) + "has been successfully added to lib")

def FindBook(self, info\_s):

for search in self.array:

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

print("Book has been founded")

print(str(search.name) + "\n" + str(search.year) + "\n" + str(search.genre))

else:

print("No one book has been founded")

def DelBook(self, b):

for search in self.array:

if b is search:

self.array.remove(b)

print("Book has been successfully deleted")

A = Lib()

name = input("Введіть назву книги")

year = int(input("Введіть дату редакції книги"))

genre = input("Введіть жанр книги")

book = Book(name, year, genre)

A.AddBook(book)

A.FindBook(Name)

A.DelBook(book)

Лістинг 1 – Програмний код до завдання 1

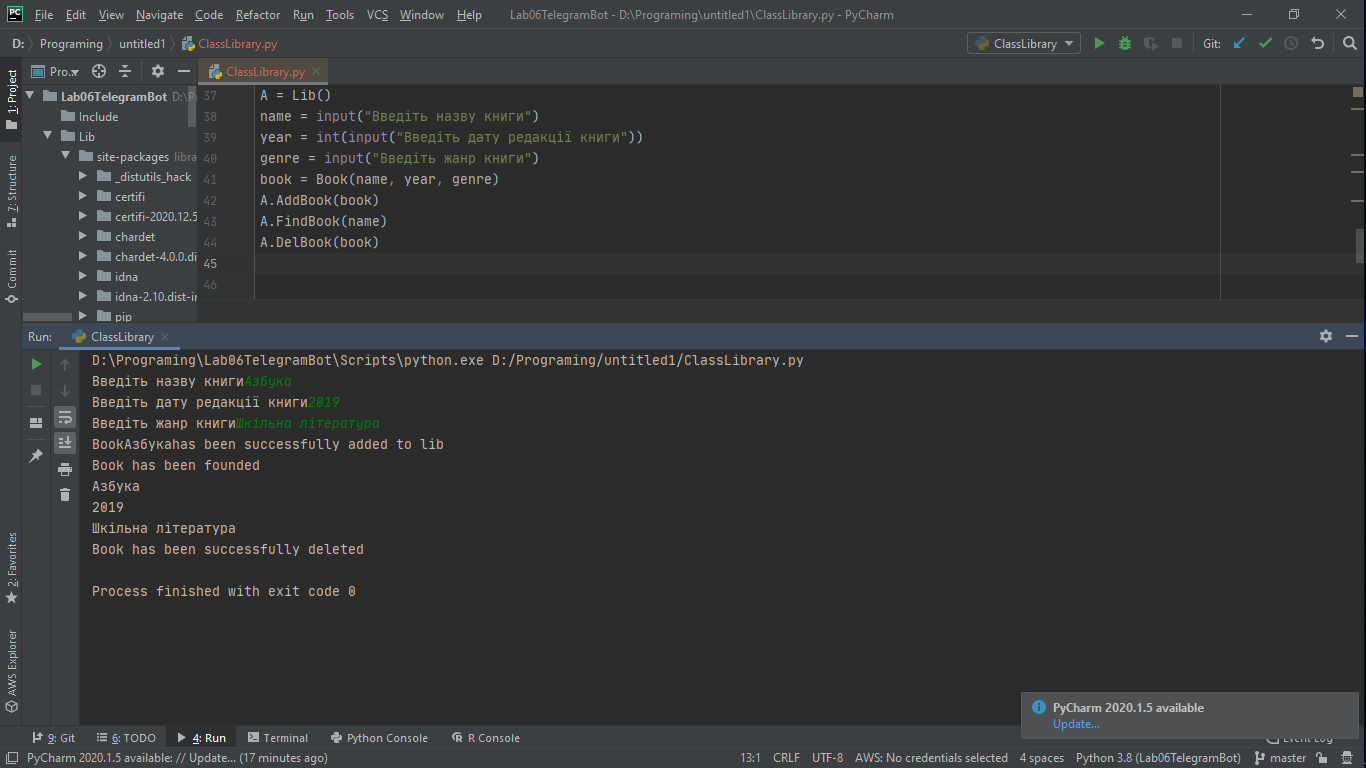


Рисунок 1 – Результат виконання завдання 1

**Завдання 2**

from random import randint

class Student:

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

if isinstance(dictionary["max\_score\_indiv"], int) and isinstance(dictionary["max\_score\_lab"], int):

if dictionary["amount\_labs"] and dictionary["score\_for\_passing"]:

self.name = name

self.max\_score\_i = dictionary["max\_score\_indiv"]

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

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

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

self.marks\_for\_labs = list()

self.mark\_for\_indiv = 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 isinstance(try\_pass, int) and isinstance(mark, int) and mark < self.max\_score\_l and self.max\_score\_l:

for i in range(self.amount\_labs):

passed\_try = randint(0, try\_pass)

if passed\_try >= 1:

self.marks\_for\_labs.append(randint(mark, self.max\_score\_l))

else:

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

return 1

else:

return 0

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

if isinstance(try\_pass, int) and isinstance(mark, int) and mark < self.max\_score\_i and self.max\_score\_i:

passed\_try = randint(0, try\_pass)

if passed\_try >= 1:

self.mark\_for\_indiv = randint(mark, self.max\_score\_i)

else:

self.mark\_for\_indiv = mark

return 1

else:

return 0

def get\_res(self):

num = self.mark\_for\_indiv

for mark in self.marks\_for\_labs:

num += mark

if num >= self.score\_part:

log\_val = True

else:

log\_val = False

return (num, log\_val)

name = input("Введіть ім'я студента")

score\_lab = input("Введіть максимальну оцінку за здачу лабораторної роботи")

score\_indiv = input("Введіть максимальну оцінку за здачу індивідуального завдання")

labs = input("Введіть кількість лабораторних робіт")

res\_score = input("Введіть кількість балів, що потрібно набрати студенту")

Stud =Student(name, max\_score\_indiv=score\_indiv, max\_score\_lab=score\_lab, amount\_labs=labs, score\_for\_passing=res\_score)

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

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

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

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

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

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

else:

print("Помилка обробки данних студента через невірно введені дані")

Лістинг 2 – Програмний код до завдання 2

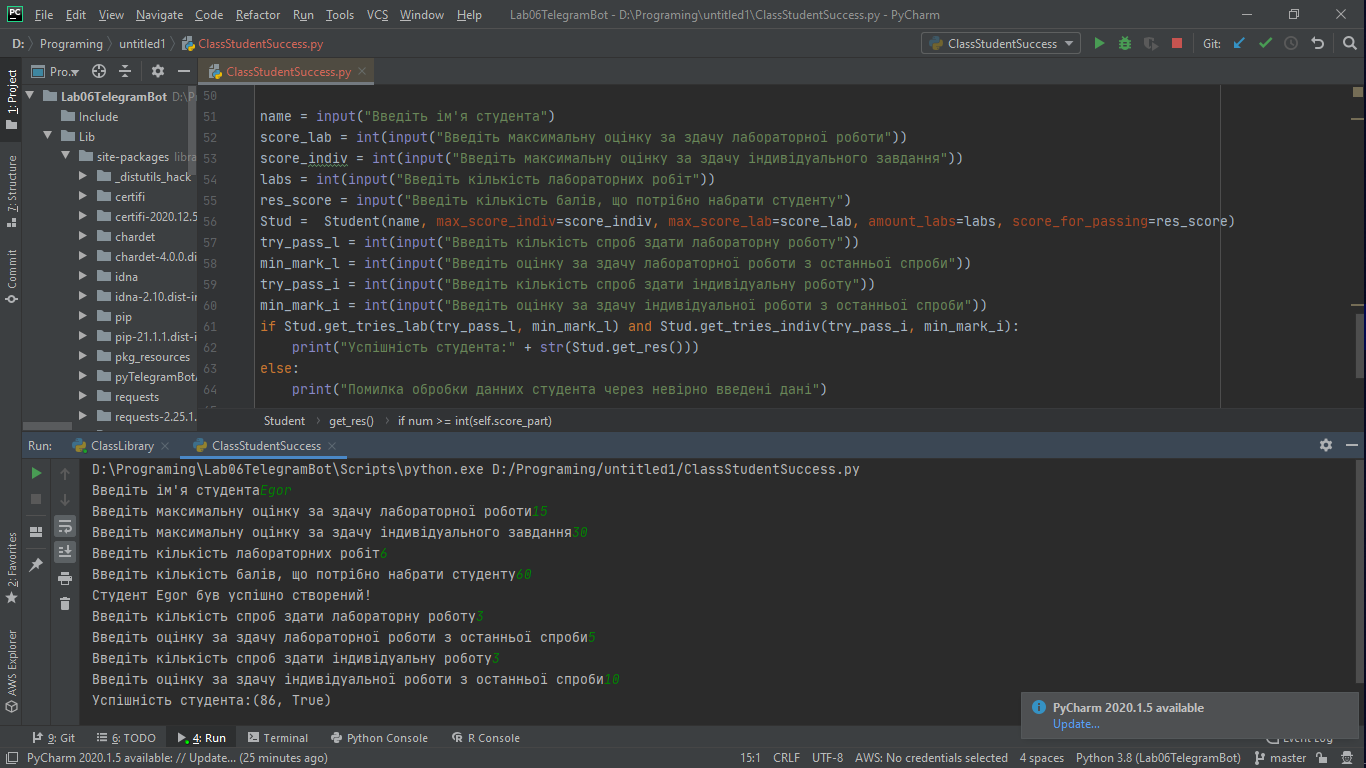


Рисунок 2 – Результат виконання завдання 2

**Завдання 3**

class ModString (str):

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

self.s = input

def substring(self):

result = False

s = self.s.lower()

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

sub = s[i:i+3]

print(sub)

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("xanaxbnaxwnax")

print(stroka.substring())

print(stroka.polindrom())

Лістинг 3 – Програмний код до завдання 3

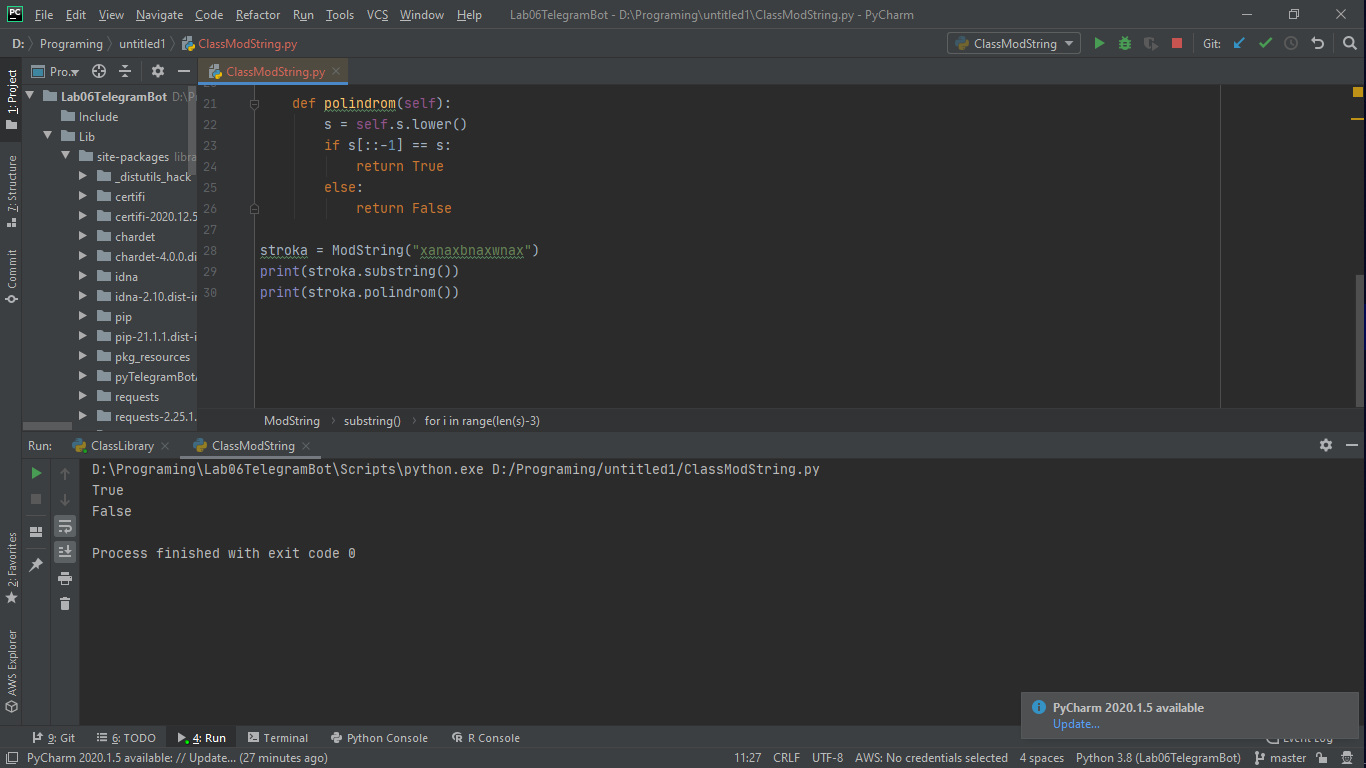


Рисунок 3 – Результат виконання завдання 3

**Завдання 4**

іmport random

class Card:

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

self.mast = random.choice(["chirva", "pika", "chresta", "bubna"])

self.num = random.choice([6, 7, 8, 9, 10, "V", "D", "K", "A"])

def show(self):

print("Mast: " + self.mast + "\n" + "Num: " + str(self.num))

class Stack:

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

self.stack = list()

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("Card from stack by index: " + str(index))

self.stack[index].show()

else:

print("Index out of range")

def showall(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:

print(elem.show())

A = Stack()

A.showall()

print("\nКарта у 5 позиції\n")

A.findbyindex(5)

print("\nШість випадкових карт\n")

A.getsixcards()

print("\n!!! NEW stack !!!\n")

A.transport()

Лістинг 4 – Програмний код до завдання 4

D:\Programing\Lab06TelegramBot\Scripts\python.exe D:/Programing/untitled1/ClassStackOfCards.py

Mast: bubna

Num: 6

Mast: pika

Num: 7

Mast: chresta

Num: K

Mast: bubna

Num: 8

Mast: chirva

Num: 7

Mast: bubna

Num: D

Mast: chirva

Num: V

Mast: pika

Num: K

Mast: pika

Num: 8

Mast: bubna

Num: D

Mast: chresta

Num: V

Mast: chirva

Num: 6

Mast: chirva

Num: 10

Mast: chresta

Num: 8

Mast: bubna

Num: 8

Mast: bubna

Num: K

Mast: bubna

Num: D

Mast: bubna

Num: K

Mast: bubna

Num: 7

Mast: chresta

Num: A

Mast: chirva

Num: 6

Mast: pika

Num: 8

Mast: pika

Num: 10

Mast: pika

Num: 9

Mast: chresta

Num: 6

Mast: pika

Num: 9

Mast: chirva

Num: K

Mast: pika

Num: V

Mast: chirva

Num: A

Mast: pika

Num: D

Mast: chirva

Num: 9

Mast: bubna

Num: D

Mast: chirva

Num: A

Mast: chresta

Num: D

Mast: chirva

Num: V

Карта у 5 позиції

Card from stack by index: 5

Mast: bubna

Num: D

Шість випадкових карт

Mast: chirva

Num: 6

Mast: chirva

Num: K

Mast: bubna

Num: K

Mast: pika

Num: 9

Mast: pika

Num: 8

Mast: pika

Num: 10

!!! NEW stack !!!

Mast: chirva

Num: 7

Mast: bubna

Num: 6

Mast: bubna

Num: 8

Mast: chirva

Num: 6

Mast: bubna

Num: D

Mast: chresta

Num: 6

Mast: bubna

Num: D

Mast: pika

Num: D

Mast: bubna

Num: 7

Mast: chirva

Num: 6

Mast: chresta

Num: 8

Mast: bubna

Num: 8

Mast: pika

Num: V

Mast: bubna

Num: D

Mast: pika

Num: 8

Mast: pika

Num: 7

Mast: bubna

Num: K

Mast: chirva

Num: A

Mast: pika

Num: K

Mast: chirva

Num: V

Mast: pika

Num: 8

Mast: chresta

Num: K

Mast: chresta

Num: A

Mast: pika

Num: 9

Mast: chirva

Num: V

Mast: chirva

Num: 9

Mast: chirva

Num: K

Mast: chresta

Num: V

Mast: chirva

Num: 10

Mast: bubna

Num: K

Mast: pika

Num: 9

Mast: bubna

Num: D

Mast: chirva

Num: A

Mast: pika

Num: 10

Mast: chresta

Num: D

Лістинг 5 – Результат виконання завдання 4

**Завдання 5**

class Dictionary:

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

self.dictionary = dict()

def addwordtranslate(self, word, translations):

if word != "" or translations != "":

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

self.dictionary[word] = translations

print("Word \"" + word + "\" has been succesfully added to dictionary")

else:

print("Word is already exist in dictionary!")

else:

print("Error of adding word to dictionary!")

def showtranslation(self, word):

if word in self.dictionary.keys():

print("Translations for word \"" + word + '\":')

print(self.dictionary[word])

else:

print("Error, no such word \"" + word + "\" in dictionary!")

D = Dictionary()

D.addwordtranslate("catch", ["поймать", "ловить", "уловить", "догнать"])

D.addwordtranslate("catch", "gwqwe")

D.addwordtranslate("parents", "родители")

D.showtranslation("parents")

D.showtranslation("four")

Лістинг 6 – Програмний код до завдання 5

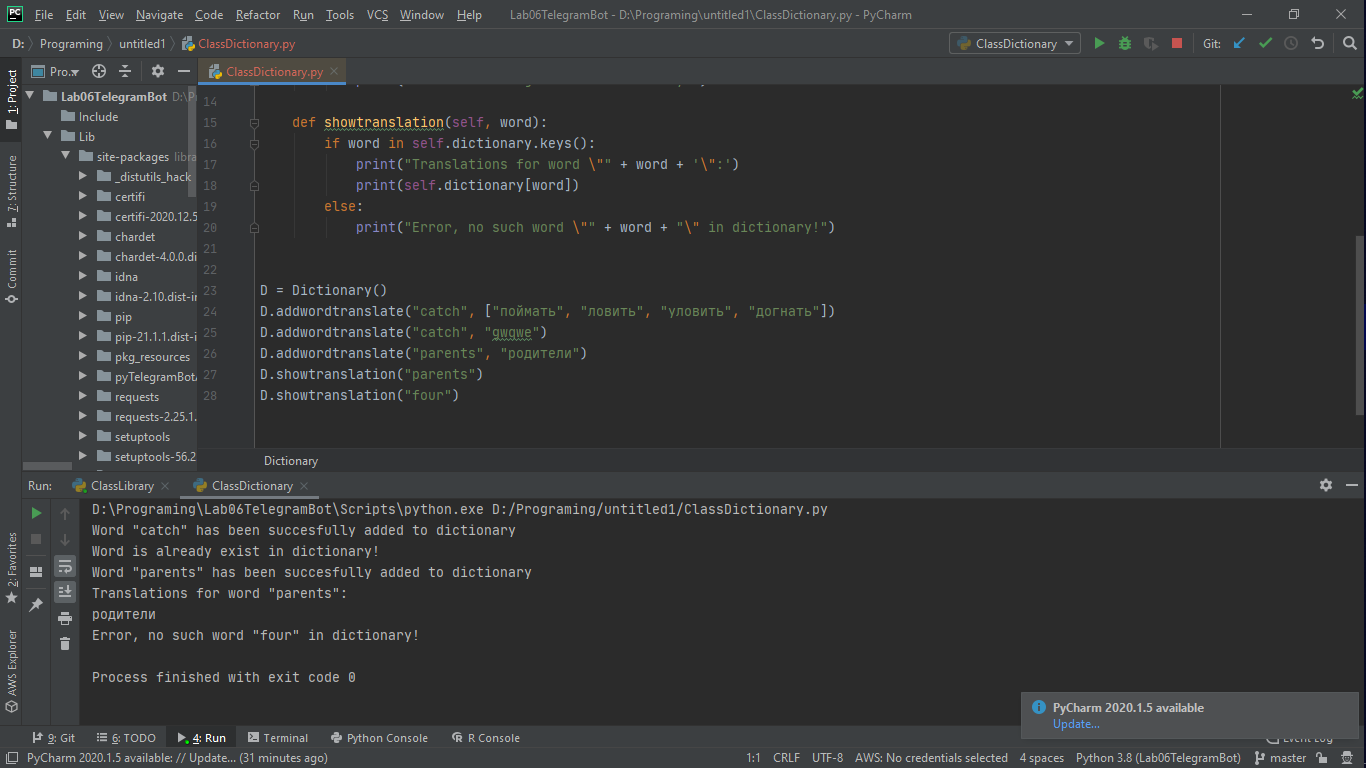


Рисунок 4 – Результат виконання завдання 5

**Завдання 6**

class Transport:

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

self.value

self.speed

self.year

self.coord

def setval(self, input):

if isinstance(input, float) or isinstance(input, int):

self.value = input

def setspeed(self, input):

if isinstance(input, int):

self.speed = input

def setyear(self, input):

if isinstance(input, int):

self.year = input

def setcoord(self, input):

if isinstance(input, str):

self.coord = input

class Car (Transport):

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

self.value = 0

self.speed = 0

self.year = 0

self.coord = ""

def get\_car\_characteristics(self):

print("\nCar characteristics\n"

+ "Value: " + str(self.value)

+ "\nSpeed: " + str(self.speed)

+ "\nYear: " + str(self.year)

+ "\nCoordinates: " + self.coord)

class Plane (Transport):

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

self.value

self.speed

self.year

self.height

self.passengers

self.coord

def \_setheight(self, input):

if isinstance(input, int):

self.height = input

def setpassengers(self, input):

if isinstance(input, int):

self.passengers = input

def \_get\_plane\_characteristics(self):

print("\nPlane characteristics\n"

+ "Value: " + str(self.value)

+ "\nSpeed: " + str(self.speed)

+ "\nYear: " + str(self.year)

+ "\nCoordinates: " + self.coord

+ "\nHeight: " + str(self.height)

+ "\nPassengers: " + str(self.passengers))

class Ship (Plane):

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

self.value

self.speed

self.year

self.port

self.passangers

self.coord

def setport(self, input):

if isinstance(input, str):

self.port = input

def get\_ship\_characteristics(self):

print("\nShip characteristics\n"

+ "Value: " + str(self.value)

+ "\nSpeed: " + str(self.speed)

+ "\nYear: " + str(self.year)

+ "\nCoordinates: " + self.coord

+ "\nPort: " + self.port

+ "\nPassengers: " + str(self.passengers))

Lancer = Car()

Lancer.setval(15000.0)

Lancer.setspeed(300)

Lancer.setyear(2004)

Lancer.setcoord("12:04:255")

Lancer.get\_car\_characteristics()

Лістинг 6 – Програмний код до завдання 6

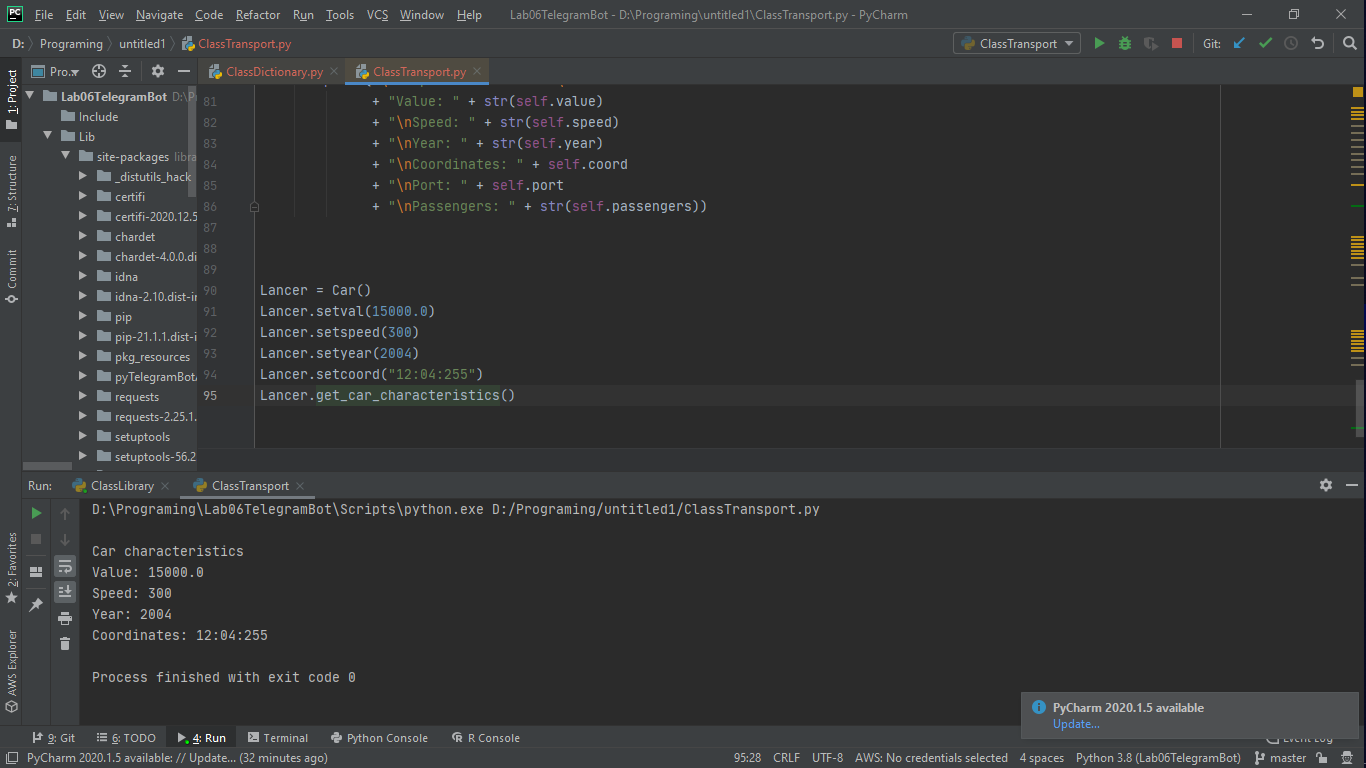


Рисунок 5 – Результат виконання завдання 6