Міністерство освіти і науки України

Державний університет ,,Житомирська політехніка”

Кафедра ФІКТ

Група:ВТ-21-1

Програмування мовою Python

Лабораторна робота №

«КЛАСИ. Ч. 2»

Виконав: Вигнич О. С.

Прийняв: Морозов Д. С.

**Мета роботи:** ознайомитися з алгоритмами послідовної (лінійної) структури, з процедурами запуску програм, які реалізують ці алгоритми на мові Python; знайомство з інтегрованим середовищем розробки – integrated development environment (IDLE).

Завдання 1: Напишіть клас Bank для опису простих операції з вашим банківським

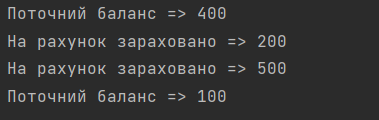
рахунком: покласти на рахунок, зняти з рахунку, переглянути рахунок.

При створенні екземпляру класу, екземпляр отримує атрибут \_\_balance з

певним значенням. Клас повинен містити методи для додавання коштів

на рахунок і знімання з рахунку, за умови, що на рахунку достатньо

коштів.   
import math  
  
  
class Bank:  
 def \_\_init\_\_(self, balance = 100):  
 self.\_\_balance = balance  
 @property  
 def balance(self):  
 return self.\_\_balance  
 def putOn(self, sum):  
 if sum > 0:  
 self.\_\_balance += sum  
 print(f"На рахунок зараховано => {sum}")  
 else: print("Сума має бути більше 0")  
 def withDraw(self, sum):  
 if math.fabs(sum) <= self.\_\_balance:  
 self.\_\_balance -= sum  
 print(f"На рахунок зараховано => {sum}")  
 else: print('Сума має бути менша наж поточний балансе')  
 def \_\_str\_\_(self):  
 return (f"Поточний баланс => {self.\_\_balance}")  
person = Bank(400)  
print(person)  
person.putOn(200)  
person.withDraw(500)  
print(person)



Завдання 2. Напишіть клас Coin, який описує монету, яку можна підкидати. При

створенні екземпляру класу, екземпляр отримує атрибут \_\_sideup зі

значенням heads або tails. У класі визначте метод toss, який випадково

визначає результат підкидання монети - орел чи решка. Створіть

екземпляр класу і виведіть на екран n підкидань монети.

import random  
  
  
class Coin:  
 def \_\_init\_\_(self):  
 self.counter = 0  
 self.\_\_sideUp = 'head'  
 def Toss(self):  
 self.counter += 1  
 arr = ['head','tail']  
 self.\_\_sideUp = random.choice(arr)  
 def \_\_str\_\_(self):  
 return self.\_\_sideUp  
myCoin = Coin()  
print(myCoin)  
myCoin.Toss()  
print(myCoin)  
myCoin.Toss()  
print(myCoin)  
print(myCoin.counter)



3. Напишіть клас Car, який надає для створених екземплярів такі атрибути

даних автомобіля: марку виготовлення автомобіля, модель автомобіля,

рік автомобіля, швидкість (початкове значення 0). Клас також повинен

мати наступні методи: accelerate (метод повинен щоразу додавати 5 до

значення атрибуту даних про швидкість), brake (метод повинен віднімати

5 від значення атрибута даних швидкості кожного разу, коли він

викликається), get\_speed (метод повинен повернути поточну швидкість).

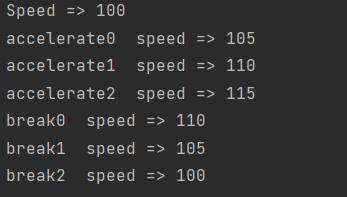
Створіть екземпляр класу Car і викличте метод accelerate п’ять разів.

Після кожного виклику методу accelerate отримайте поточну швидкість

автомобіля і надрукуйте її значення. Потім викличте метод brake п’ять

разів. Після кожного виклику методу brake отримайте поточну швидкість

автомобіля та надрукуйте її значення.class Car:  
 def \_\_init\_\_(self, brend, model, speed = 0):  
 self.brend = brend  
 self.model = model  
 self.speed = speed  
 def accelerate(self):  
 self.speed += 5  
 def brake(self):  
 self.speed -= 5  
 def getSpeed(self):  
 return self.speed  
  
mycar = Car("Tesla", "x", 100)  
print(f"Speed => {mycar.getSpeed()}")  
for i in range(3):  
 mycar.accelerate()  
 print(f"accelerate{i} speed => {mycar.getSpeed()}")  
for i in range(3):  
 mycar.brake()  
 print(f"break{i} speed => {mycar.getSpeed()}")



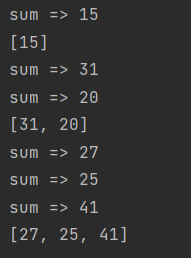
4. Напишіть клас Dog, який має три атрибути класу: mammal (ссавець), nature (характер) і breed (порода), та два атрибути ексземпляра: name (кличка) і age (вік). Створіть екземпляри трьох нових собак, кожна з яких різного віку. Визначте у класі Dog метод для виведення значень атрибутів екземпляру - імені та віку конкретної собаки. За потреби, додайте кілька інших методів, які визначають поведінку собаки (подавання голосу тощо). Напишіть кілька класів, які унаслідуються від батьківського класу Dog, що описують конкретні породи собак. Визначте для цих класів атрибути nature і breed відповідно, включіть у класи по одному методу, що визначає поведінку конкретної породи собаки. Створіть батьківський клас Pets, що створює список ваших домашніх улюбленців. У підсумку, надрукуйте інформацію про ваших домашніх тварин, на зразок, як у вихідних даних.

class Pets:  
 pets = []  
 def Show(self):  
 for i in self.pets:  
 print(i)  
class Dog(Pets):  
 nature = ""  
 breed = ""  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
 self.pets.append(self)  
 def \_\_str\_\_(self):  
 return (f"name:{self.name} age {self.age}")  
  
 def behavior(self):  
 return "Gaw"  
class Akita(Dog):  
 breed = "Akita"  
 nature = "kind"  
 def behavior(self):  
 return "Gaaw"  
class Terrier(Dog):  
 breed = "Terrier"  
 nature = "angry"  
 def behavior(self):  
 return "Gaaaw"  
  
dog1 = Akita("qwe",5)  
dog2 = Terrier("asd",6)  
dog1.Show()



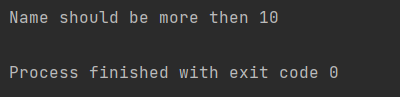
5. Дано послідовність цілих чисел. Необхідно її обробити і вивести на екран суму першої п’ятірки чисел із цієї послідовності, потім суму другої п’ятірки, і т. д. Але послідовність не дається відразу загалом. З плином часу до вас надходять її послідовні частини. Наприклад, спочатку перші три елементи, потім наступні шість, потім наступні два і т. д. Реалізуйте клас Buffer, який буде накопичувати в собі елементи послідовності і виводити суму п’ятірок послідовних елементів у міру їх накопичення. Однією з вимог до класу є те, що він не повинен зберігати в собі більше елементів, ніж йому дійсно необхідно, тобто, він не повинен зберігати елементи, які вже увійшли в п’ятірку, для якої була виведена сума. Клас повинен мати наступний вигляд

class Buffer:  
 def \_\_init\_\_(self):  
 self.arr = []  
 self.last = 0  
 self.arrSum = []  
 def add(self, \*a):  
 i = 0  
 self.arrSum.clear()  
 arr = self.arr[self.last:len(self.arr)]  
 arr.extend(a)  
 while i <= len(arr):  
 if i+4 < len(arr):  
 self.Summ(arr[i:i+5])  
 self.arr[len(self.arr):] = arr[i:i+5]  
 i += 5  
 else:  
 self.last = len(self.arr)  
 self.arr[len(self.arr):] = arr[i:]  
 break  
 def Summ(self, a):  
 sum = 0  
 for i in a:  
 sum += i  
 print(f"sum => {sum}")  
 self.arrSum.append(sum)  
 def get\_current\_part(self):  
 return self.arrSum  
  
myByffer = Buffer()  
myByffer.add(1,2,3,4,5,6,7,8,9)  
print(myByffer.get\_current\_part())  
myByffer.add(1,2,3,4,5,6,7,8,9)  
print(myByffer.get\_current\_part())  
myByffer.add(1,2,3,4,5,6,7,8,9,1,22,1,2,3,4)  
print(myByffer.get\_current\_part())



6. Напишіть клас-виняток, на основі вбудованого в Python класу ValueError(). Клас буде представляти перевірку певного імені на основі його довжини. Якщо довжина введеного імені є меншою 10, то має генеруватися виняток як у вихідних даних. У інших випадках нічого не виводиться.

class NameLengthError(Exception):  
 pass  
  
def average(name):  
 if len(name) <= 10:  
 raise NameLengthError('Name should be more then 10')  
 return name  
  
def CheckName(name):  
 try:  
 value = average(name)  
 except NameLengthError:  
 print('Name should be more then 10')  
 else:  
 return value  
CheckName("qwecvnbbjjj")



7. Напишіть один клас для перетворення десяткового числа на число в римській системі числення. І ще один клас для перетворення числа з римської системи числення у десяткове число.

class IntoRoman:  
 ones = ["", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"]  
 tens = ["", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"]  
 hunds = ["", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"]  
 thous = ["", "M", "MM", "MMM", "MMMM"]  
 def \_\_init\_\_(self, numb):  
 self.\_\_numb = self.arabic\_to\_roman(numb)  
 def arabic\_to\_roman(self, data):  
 t = self.thous[data // 1000]  
 h = self.hunds[data // 100 % 10]  
 te = self.tens[data // 10 % 10]  
 o = self.ones[data % 10]  
 return t + h + te + o  
 def \_\_str\_\_(self):  
 return self.\_\_numb  
  
class IntoArabic:  
 rule\_add = {  
 'I': 1,  
 'V': 5,  
 'X': 10,  
 'L': 50,  
 'C': 100,  
 'D': 500,  
 'M': 1000,  
 }  
 rule\_div = {  
 ('I', 'V'): 3,  
 ('I', 'X'): 8,  
 ('X', 'L'): 30,  
 ('X', 'C'): 80,  
 ('C', 'D'): 300,  
 ('C', 'M'): 800,  
 }  
 def \_\_init\_\_(self, numb):  
 self.\_\_numb = self.roman\_to\_arabic(str(numb))  
 def roman\_to\_arabic(self, data):  
 number = 0  
 prev\_literal = None  
 for literal in data:  
 if prev\_literal and self.rule\_add[prev\_literal] < self.rule\_add[literal]:  
 number += self.rule\_div[(prev\_literal, literal)]  
 else:  
 number += self.rule\_add[literal]  
 prev\_literal = literal  
 return number  
 def \_\_str\_\_(self):  
 return str(self.\_\_numb)  
  
rez = IntoRoman(29)  
print(rez)  
rez = IntoArabic(rez)  
print(rez)

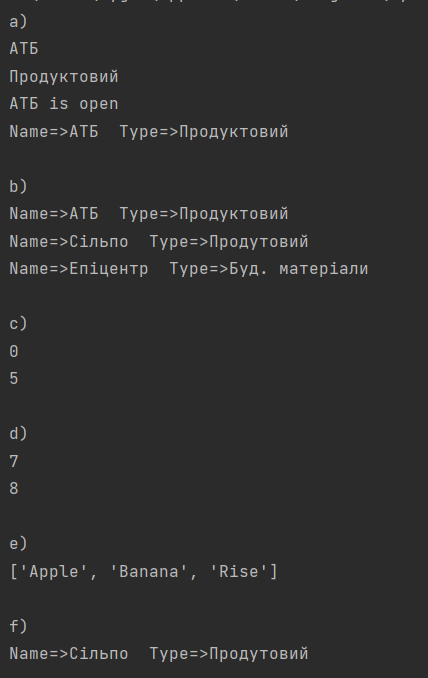


8. Онлайн-магазин.

Клас Shop

class Shop:  
 def \_\_init\_\_(self, name, type, units = 0):  
 self.number\_of\_units = units  
 self.shop\_name = name  
 self.shop\_type = type  
 def describe\_shop(self):  
 print(f"Name=>{self.shop\_name} Type=>{self.shop\_type}")  
 def open\_shop(self):  
 print(f"{self.shop\_name} is open")  
 def set\_number\_of\_units(self, numb):  
 self.number\_of\_units = numb  
 def increment\_number\_of\_units(self):  
 self.number\_of\_units += 1

from shop import Shop  
  
class Discount(Shop):  
 def \_\_init\_\_(self, arr):  
 self.discount\_products = arr  
  
 def get\_discounts\_ptoducts(self):  
 print(self.discount\_products)  
# A)  
print("a)")  
store = Shop("АТБ","Продуктовий")  
print(store.shop\_name)  
print(store.shop\_type)  
store.open\_shop()  
store.describe\_shop()  
  
store2 = Shop("Сільпо", 'Продутовий')  
store3 = Shop("Епіцентр", 'Буд. матеріали')  
# B)  
print("\nb)")  
store.describe\_shop()  
store2.describe\_shop()  
store3.describe\_shop()  
  
# C)  
print("\nc)")  
store = Shop("АТБ","Продуктовий")  
print(store.number\_of\_units)  
store.number\_of\_units = 5  
print(store.number\_of\_units)  
  
# D)  
print("\nd)")  
store.set\_number\_of\_units(7)  
print(store.number\_of\_units)  
store.increment\_number\_of\_units()  
print(store.number\_of\_units)  
  
# E)  
print("\ne)")  
store\_discount = Discount(["Apple","Banana","Rise"])  
store\_discount.get\_discounts\_ptoducts()  
  
# F)  
print("\nf)")  
all\_store = Shop("Сільпо", 'Продутовий')  
all\_store.describe\_shop()



9. Облік користувачів на сайті.

Class User

class User:  
 def \_\_init\_\_(self, first\_name, last\_name, nickname, login\_attempts = 0):  
 self.first\_name = first\_name  
 self.last\_name = last\_name  
 self.nickname = nickname  
 self.login\_attempts = login\_attempts  
 def describe\_user(self):  
 print(f"Name:{self.first\_name} Surname:{self.last\_name} Nackname:{self.nickname}")  
 def greeting\_user(self):  
 print(f"Hello {self.nickname}")  
 def increment\_login\_attempts(self):  
 self.login\_attempts += 1  
 def reset\_login\_attempts(self):  
 self.login\_attempts = 0

class Privileges i Admin

from user import User  
class Privileges:  
 privileges = ["Allowed to add message", "Allowed to delete users", "Allowed to ban users"]  
  
 def show\_privileges(self):  
 print(self.privileges)  
class Admin(User):  
 privileges = ["Allowed to add message", "Allowed to delete users", "Allowed to ban users"]  
 def show\_privileges(self):  
 print(self.privileges)  
 priv = Privileges()

from user import User  
from PrivAndAdmin import Admin  
  
# A)  
print("a)")  
user1 = User("User1(First)", "User1(last)","User1(nick)")  
user2 = User("User2(First)", "User2(last)","User2(nick)")  
user1.describe\_user()  
user1.greeting\_user()  
user2.describe\_user()  
user2.greeting\_user()  
  
# B)  
print("\nb)")  
user = User("User(First)", "User(last)","User(nick)", 1)  
print(user.login\_attempts)  
user.increment\_login\_attempts()  
user.increment\_login\_attempts()  
user.increment\_login\_attempts()  
print(user.login\_attempts)  
user.reset\_login\_attempts()  
print(user.login\_attempts)  
  
# C)  
print("\nc)")  
admin = Admin("Admin(First)", "Admin(last)","Admin(nick)", 1)  
admin.show\_privileges()  
  
# D)  
print("\nd)")  
admin = Admin("Admin(First)", "Admin(last)","Admin(nick)", 1)  
admin.priv.show\_privileges()

