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.

					ІРТР.420001.123-ЗЛ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Вигнич О.С..			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів
Перевір.		Морозов Д. С.					
Керівник						ФІКТ, гр. ВТ-21-1	
Н. контр.							
Затверд.							

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)
```

```
Поточний баланс => 400
На рахунок зараховано => 200
На рахунок зараховано => 500
Поточний баланс => 100
```

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)
```

```
head
head
tail
2
```

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()}")
```

```
Speed => 100
accelerate0  speed => 105
accelerate1  speed => 110
accelerate2  speed => 115
break0  speed => 110
break1  speed => 105
break2  speed => 100
```

4. Напишіть клас Dog, який має три атрибути класу: mammal (ссавець), nature (характер) і breed (порода), та два атрибути екземпляра: 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()
```

```
name:qwe age 5

name:asd age 6
```

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())
```

```
sum => 15
[15]
sum => 31
sum => 20
[31, 20]
sum => 27
sum => 25
sum => 41
[27, 25, 41]
```

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")
```

```
Name should be more then 10


Process finished with exit code 0
```

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)
```

```
XXIX
29
```

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()
```

```
a)
АТБ
Продуктовий
АТБ is open
Name=>АТБ  Type=>Продуктовий

b)
Name=>АТБ  Type=>Продуктовий
Name=>Сільпо  Type=>Продутовий
Name=>Епіцентр  Type=>Буд. матеріали

c)
0
5

d)
7
8

e)
['Apple', 'Banana', 'Rise']

f)
Name=>Сільпо  Type=>Продутовий
```

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()
a)
Name:User1(First) Surname:User1(last) Nackname:User1(nick)
Hello User1(nick)
Name:User2(First) Surname:User2(last) Nackname:User2(nick)
Hello User2(nick)

b)
1
4
0

c)
['Allowed to add message', 'Allowed to delete users', 'Allowed to ban users']

d)
['Allowed to add message', 'Allowed to delete users', 'Allowed to ban users']
```