Лабораторна роботи № 9

Елементи ООП на Python. Розробка класів. Інкапсуляція. Наслідування і поліморфізм.

Мета роботи: *ознайомлення з основними парадигмами об'єктно-орієнтованого програмування та їх реалізацією в Python.*

Короткі теоретичні відомості

Python підтримує об'єктно-орієнтовану парадигму програмування вже в основі. Майже все в Python є об'єктом зі своїми властивостями та методами. Клас в ООП певною мірою схожий на конструктор об'єктів або «схему» для створення об'єктів. Для створення класу, використовується ключове слово class, наприклад:

```
class MyClass:
   x = 5
```

Ці інструкції створюють клас з назвою ***MyClass*** із властивістю ***x***. Далі можна використовувати клас під назвою ***MyClass*** для створення об'єктів:

```
p1 = MyClass()
print(p1.x)
```

В якості конструктора класу, що дозволяє створювати об'єкти в Python використовують вбудовану функцію ***__init__()***.

Функція під назвою ***__init__()*** виконується під час створення об'єкта, її використовують для того, щоб оголосити та ініціалізувати властивості об'єкта. Створимо клас під назвою ***Person***, скориставшись функцією ***__init__()***, щоб призначити значення імені та віку:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
p1 = Person("John", 36)
print(p1.name)
print(p1.age)
```

В описі класу можна розмістити декілька реалізацій функції ***__init__()*** з різними кількостями вхідних параметрів, проте працювати буде лише одна з них - та, визначення якої розташоване в коді класу найнижче.

Об'єкти також можуть містити методи. Методи в об’єктах — це функції, які належать об’єкту.

Параметр ***self*** є посиланням на поточний екземпляр класу(об'єкт) та використовується для доступу до змінних, які належать об'єктові класу.

Її не обов’язково називати ***self***, ви можете називати її як завгодно, але вона має бути першим параметром будь-якої функції в класі:

Наприклад, використаємо слова ***body*** та ***p1*** замість self :

```
class Person:
    def __init__(body, name, age):
        body.name = name
        body.age = age
    def myfunc(p1):
        print("Hello my name is " + p1.name)
p = Person("John", 36)
p.myfunc()
```

Функція ***__str__()*** контролює те, як повертаються дані об’єкта у вигляді рядка при виводі на друк.

Спадкування дозволяє визначити клас, який успадковує всі методи та властивості від іншого класу. Батьківський клас — це клас, який успадковується, також називається базовим класом. Дочірній клас — це клас, який успадковує від іншого класу, який також називають похідним класом.

Щоб створити клас, який успадковує функції від іншого класу у Python, передають батьківський клас як параметр під час створення дочірнього класу. Наприклад, клас ***Student***, який успадкує властивості та методи від класу ***Person***:

```
class Student(Person):
    pass
```

Коли ми додаємо функцію ***__init__()*** в дочірній клас, він перестає успадковувати батьківську функцію ***__init__()***, бо функція дочірнього елемента перекриває успадкувану функцію.

Щоб зберегти успадкування батьківської ***__init__()*** функції, слід додати явний виклик батьківської init-функції, наприклад:

```
class Student(Person):
    def __init__(self, fname, lname):
        Person.__init__(self, fname, lname)
```

Python також має функцію ***super()***, яка змусить дочірній клас успадкувати всі методи та властивості свого батька:

Наприклад,

```
class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
```

Використовуючи функцію super(), не потрібно використовувати ім'я батьківського елемента, він автоматично успадковує методи та властивості свого батька.

Окрім згаданих вище вбудованих функцій ***__init__(self[, ...])*** та ***__str__(self)*** для класів у Python передбачено декілька десятків інших вбудованих функцій (деколи їх називають «магічними методами»), кожна з яких викликається інтерпретатором для екземляру класу в тому чи іншому контексті. Наприклад функція ***__del__(self)*** викликається при видаленні об'єкта збирачем сміття а ***__format__(self, format_spec)*** – якщо об'єкт передається функції format.

Зокрема, для об'єктів можна висзначити власні правила порівняння, реалізувавши в класі функціїї ***__lt__(self, other)***, ***__le__(self, other)***, ***__eq__(self, other)***, ***__ne__(self, other)***, ***__gt__(self, other)***, ***__ge__(self, other)***. При цьому:

вираз ***x < y*** викликатиме функцію ***x.__lt__(y)***;

вираз ***x ≤ y*** викликатиме функцію ***x.__le__(y)***;

вираз ***x == y*** викликатиме функцію ***x.__eq__(y)***;

вираз ***x != y*** викликатиме функцію ***x.__ne__(y)***;

вираз ***x > y*** викликатиме функцію ***x.__gt__(y)***;

вираз ***x ≥ y*** викликатиме функцію ***x.__ge__(y)***.

Подібним чином можна реалізувати виконання з об'єктами класу арифметичних операцій, реалізувавши відповідні функції, враховуючи, що:

__add__(self, other) для додавання (x + y);

__sub__(self, other) для віднімання (x - y);

__mul__(self, other) для множення (x * y);

__truediv__(self, other) для ділення (x / y);

__floordiv__(self, other) для цілочисельного ділення (x // y);

__mod__(self, other) для обчислення остачі від ділення (x % y);

__pow__(self, other[, modulo]) для піднесення до степеня.

Для скорочених арифметичних операторів з присвоєння визначено аналогічні функції, ім'я яких починається на «***i***», наприклад ***__iadd__(self, other)*** для x += y, ***__isub__(self, other)*** для x - = y, ***__imul__(self, other)*** для x *=y, і т.д.

Завдання

Описати власний тип даних з використанням класу згідно варіанта. Передбачити потрібні властивості класу та функції для доступу до них. Реалізувати функції __init()__ та __str()__ та функції-члени, вказані у варіанті завдання. Скласти програму, яка в діалоговому режимі пропонує користувачеві ввести дані для екземплярів, та вибрати дію, яку потрібно виконати.

Варіанти завдань

1. Клас ***вектор в тривимірному просторі***. Визначити операції: додавання двох векторів (оператор "+"), скорочений оператор присвоєння з додаванням іншого вектора (оператор "+="), скалярний добуток двох векторів (оператор "*"), обчислення довжини вектора у вигляді дійсного числа (), отримання протилежного вектора (оператор "~").

2. Клас ***раціональне число***. Визначити операції: віднімання двох раціональних чисел (оператор "-"), ділення двох раціональних чисел (оператор "/"), піднесення раціонального числа до степеня (оператор "**"), оператор декременту (оператор "--"), скорочений оператор присвоєння з додаванням (оператор "+=").

3. Клас ***квадратна матриця 3x3***. Визначити операції: додавання двох матиць (оператор "+"), віднімання двох матриць (оператор "-"), добуток двох матриць (оператор "*"), множення матриці на число зліва (оператор "*"), отримання оберненої (оператор "~").

4. Клас ***комплексне число***. Визначити операції: множення двох комплексних чисел (оператор “*”), додавання двох комплексних чисел (оператор “+”), порівняння модуля комплексного числа з дійсним числом на відношеннями «менше» «більше» (оператори “<”, “>”), множення комплексного числа на дійсне число (оператор “*”).

5. Клас ***раціональне число***. Визначити операції: множення двох раціональних чисел (оператор “*”), ділення двох раціональних чисел (оператор “/”), додавання до раціонального числа цілого числа (оператор “+”), оператор «менше» для порівняння двох раціональних чисел (оператор “<”), скорочений оператор присвоєння з відніманням (оператор “-=”).

6. Клас ***вектор в тривимірному просторі***. Визначити операції: віднімання двох векторів (оператор “-”), скорочений оператор присвоєння з множенням вектора на число (оператор “*=”), векторний добуток двох векторів (оператор “*”), множення вектора на число (оператор “*”), отримання протилежного вектора (оператор “унарний міну -”).

7. Клас ***вектор в тривимірному просторі***. Визначити операції: додавання двох векторів (оператор “+”), скорочений оператор присвоєння з відніманням іншого вектора (оператор “-=”), векторний добуток двох векторів (оператор “*”), ділення вектора на число (оператор “/”), збільшення усіх координат вектора на 1 (оператор інкременту “++”).

8. Клас ***раціональне число***. Визначити операції: віднімання двох раціональних чисел (оператор “-”), додавання двох раціональних чисел (оператор “+”), множення раціонального на ціле (оператор “*”), оператор «більше» для порівняння двох раціональних чисел (оператор “>”), скорочений оператор присвоєння з діленням (оператор “/=”).

9. Клас ***квадратна матриця 4x4***. Визначити операції: додавання двох матиць (оператор “+”), віднімання двох матриць (оператор “-”), поелементного множення двох матриць (оператор “*”), поелементного ділення матриць (оператор “/”), отримання транспонованої матриці (оператор “~”).

10. Клас ***множина цілих чисел***. Визначити операції: перетину двох множин (оператор “*”), об’єднання двох множин (оператор “+”), різниці двох множин (оператор “-”), включення цілого числа в множину (оператор “+=”), виключення елемента з множини (оператор “-=”).

11. Клас ***раціональне число***. Визначити операції: множення двох раціональних чисел (оператор “*”), додавання двох раціональних чисел (оператор “+”), порівняння раціонального числа з цілим числом на відношення «менше» (оператор “<”), оператор постфіксного інкременту (оператор “++”), оператор «менше» для порівняння двох раціональних чисел (оператор “<”).

12. Клас ***прямокутний паралелепіпед*** (зберігає розміри: ширина, висота довжина). Визначити оператори: інкременту (оператор “++”) та декременту (оператор “--”) шляхом додавання/віднімання одиниці від усіх, розмірів паралелепіпеда та усі оператори порівняння (“>”, “<”, “<=”, “>=”, “= =”, “!=”) за об’ємом, тобто більшим вважається той паралелепіпед, об’єм якого більший.

13. Клас ***комплексне число***. Визначити операції: віднімання двох комплексних чисел (оператор “-”), ділення двох комплексних чисел (оператор “/”), піднесення комплексного числа до степеня (оператор “**”), скорочений оператор присвоєння з множенням (оператор “*=”), скорочений оператор присвоєння з додаванням (оператор “+=”).

14. Клас ***комплексне число***. Визначити операції: віднімання двох комплексних чисел (оператор “-”), додавання двох комплексних чисел (оператор “+”), множення комплексного на дійсне (оператор “*”), , скорочені оператори присвоєння з множенням та діленням комплексних чисел (оператори “*=” і “/=”).

15. Клас ***раціональне число***. Визначити операції: порівняння двох раціональних чисел «більше», «менше», «рівне» (оператори “>”, “<”, “= =”), скорочені оператори присвоєння для чотирьох арифметичних операцій (оператори “+=”, “-=”, “*=”, “/=”).

16. Клас **звичайний дріб**. Визначити операції: додавання двох дробів (оператор “+”), множення двох дробів (оператор “*”), перетворення дробу на дійсне число (float), додавання та віднімання з присвоєнням (оператори “+=” та “-=”), отримання оберненого дробу (оператор “~”) та перевірки двох дробів на рівність.

Приклад виконання завдання

Варіант № 16. Умова задачі: Створити клас **звичайний дріб**. Визначити операції: додавання двох дробів (оператор “+”), множення двох дробів (оператор “*”), перетворення дробу на дійсне число (float), додавання та віднімання з присвоєнням (оператори “+=” та “-=”), отримання оберненого дробу (оператор “~”) та перевірки двох дробів на рівність.

Оголосимо клас з полями __num__ та __denum__ для чисельника та знаменника дробу відповідно. Для простоти реалізації арифметичних операцій з дробами зручно виконувати скорочення дробу перед поверненням результату, для цього реалізуємо в класі допоміжну функцію reduce. Також для скорочення дробів потрібно знаходити найбільший спільний дільник, для цього реалізуємо окрему фукцію поза класом.

Код програми може бути таким:

```
import os
def nsd(n,k):
    d = n % k
    if d != 0:
        return nsd(k,d)
    else:
        return k
class fraction:
    def reduce(self):
        d = nsd(self.__num__, self.__denom__)
        self.__num__ //= d
        self.__denom__ //= d
        # чисельник і знаменник дпробу діляться на
        # на свій спільний дільник націло, цілочисельне
        # ділення тут використвне для того, щоб
        # зберегти цілочисельний тип даних

    def __init__(self, num, denom):
        self.__num__ = num
        self.__denom__ = denom
    def __add__(self, other):
        a = self.__num__*other.__denom__ + \
                self.__denom__*other.__num__
        b = self.__denom__*other.__denom__
        result = fraction(a,b)
        result.reduce()
        return result
    def __mul__(self, other):
        a = self.__num__*other.__num__
        b = self.__denom__*other.__denom__
```

```
        result = fraction(a,b)
        result.reduce()
        return result
    def __iadd__(self, other):
        a = self.__num__ * other.__denom__ + \
                    self.__denom__ * other.__num__
        self.__denom__ = self.__denom__ * other.__denom__
        self.__num__ = a
        self.reduce()
        return self

    def __isub__(self, other):
        a = self.__num__ * other.__denom__ - \
                        self.__denom__ * other.__num__
        self.__denom__ = self.__denom__ * other.__denom__
        self.__num__ = a
        self.reduce()
        return self
    def __invert__(self):
        return fraction(self.__denom__, self.__num__)

    def __float__(self):
        return self.__num__/self.__denom__

    def __eq__(self, other):
        # копії об'єктів
        a = fraction(self.__num__, self.__denom__)
        b = fraction(other.__num__, other.__denom__)
        a.reduce()
        b.reduce()
```

```
        return a.__num__ == b.__num__ and \
                    a.__denom__ == b.__denom__
    def __str__(self):
        return f'{self.__num__} / {self.__denom__}'

# Основний код
n1 = int(input('Введіть чисельник першого дробу: '))
d1 = int(input('Введіть знаменник першого дробу: '))
n2 = int(input('Введіть чисельник другого дробу: '))
d2 = int(input('Введіть знаменник другого дробу: '))
a = fraction(n1, d1)
b = fraction(n2, d2)
a.reduce()
b.reduce()
while(True):
    print(f'a = {a},  b = {b}')
    print('Оберіть дію для перевірки:\n',
          '1 - a + b\n', '2 - a * b\n', '3 - a += b\n',
          '4 - a -= b\n', '5 - a == b\n',
          '6 - обернений до a\n', '8 - завершити',
          '7 - дріб b у вигляді десяткового дробу\n')
    choise = int(input())
    if choise == 8: break
    if choise == 1:
        c = a + b
        print(f'{a} + {b} = {c}')
    if choise == 2:
        c = a * b
        print(f'{a} * {b} = {c}')
    if choise == 3:
```

```
        a += b
        print(f'a = {a}, b = {b}')
    if choise == 4:
        a -= b
        print(f'a = {a}, b = {b}')
    if choise == 5:
        if a == b:
            print(f'{a} == {b}')
        else:
            print(f'{a} != {b}')
    if choise == 6:
        c = ~a
        print(f'Обернений до {a} дріб: {c}')
    if choise == 7:
        print(f'{b} =', '%10.5f'%b)
    os.system('pause')
```

Запитання для самоконтролю

1. Який синтаксис оголошення класу в ***Python***?
2. Яке призначення функції ***__init__*** в класах?
3. Для чого використовується функція ***__str__***?
4. Що отримують в якості пешого параметру методи класів в ***Python***?
5. Чи має ***Python*** засоби для перевизначення операторів? Які?
6. Які функції використовуються для виконання арифметичних операторів з екзкмплярами класу?
7. Які функції дозволяють реалізувати операції порівняння об’єктів класу?