Програмування

ТЕМА 14. НАСЛІДУВАННЯ ТА АГРЕГУВАННЯ

Класи та атрибути

При проектуванні класів важливо розрізняти класи та атрибути.

Різниця не завжди є однозначною та може залежати від задачі, що розглядається.

Якщо сутність має внутрішню структуру та поведінку, то вона є класом.

Якщо сутність є атомарною, то вона є атрибутом.

Класи та модулі

Інше питання, яке виникає при побудові програм, що містять класи та модулі: чи треба у певному випадку оформлювати клас або більш доцільно використати модуль.

І класи, й модулі призначені для того, щоб об’єднувати описи функцій (методів) та дані.

Але між ними існує фундаментальна різниця.

Усі змінні, які описані у модулі, існують тільки в одному екземплярі,

В той час, як клас може породжувати багато об’єктів (екземплярів класу).

Тому, якщо відомо, що деяка сутність у програмі існує (має існувати) тільки в одному екземплярі, то краще використати модуль, інакше - клас.

Модулі також слугують контейнерами для опису класів (містять описи одного або декількох класів.

Наслідування та агрегування

Наслідування не є єдиним відношенням між класами.

Не менш важливим є відношення агрегації.

Кажуть, що між класами існує відношення **агрегації**, якщо об’єкти одного класу включають в себе об’єкти іншого класу.

Це включення може бути фізичним (автомобіль – колесо) або логічним (вікно – шрифт).

У першому випадку кажуть про **композицію**, а у другому – про **агрегацію**.

У Python агрегація реалізується вказанням об’єктів одного класу в якості полів іншого класу.

Приклад: обчислення трикутників з максимальною площею та периметром

Трикутник задається трьома точками.

Дано декілька трикутників.

Знайти серед них трикутник з найбільшою площею та трикутник з найбільшим периметром.

Для розв’язання задачі опишемо класи: Point2 – точка площини, Segment – відрізок,Triangle – трикутник. Трикутник та відрізок містять точки, тобто об’єкти класу Point2.

Стандартні контейнери

Контейнерами називають класи, що призначені для зберігання та обробки сукупностей об’єктів інших класів.

Python має значну кількість стандартних контейнерів.

Серед них рядки, списки, кортежі, множини, словники, які ми розглянули раніше.

Але у Python є й інші стандартні контейнери, більшість з яких зібрано у модулі collections.

З цим модулем ми вже знайомі за іменованим кортежами (namedtuple).

Стандартні контейнери.2

Розглянемо більш докладно такі контейнери:

- Counter
- defaultdict
- OrderedDict
- deque

Counter

Клас Counter призначено для підрахунку кількості входжень однакових елементів з деякої послідовності.

Для створення об’єкту цього класу треба викликати конструктор:

c = Counter(seq)

- операції для Counter:

+, -, |, &

- відношення

in, not in

- інструкції

print

Методи для об’єктів класу Counter

c.elements() – повертає послідовність усіх елементів у довільному порядку, але кожний елемент повторюється стільки разів, скільки він входить до c.

c.most_common([n]) – повертає n елементів, що входять до c найчастіше.

Приклад: слово, яке повторюється найчастіше

Знайти слово, яке найчастіше повторюється у заданому файлі

defaultdict

Клас defaultdict призначено для створення та обробки словників, у які автоматично додаються елементи, якщо їх не існує.

Для створення такого словника потрібно набрати

d = defaultdict(cls)

де cls – це клас елементів словника.

defaultdict.2

Після створення словника звернення до його елемента d[x] поверне значення, якщо елемент існує.

Якщо ж елемента у словнику немає, то буде викликаний конструктор для створення об’єкту класу cls, цей об’єкт буде додано до словника з ключем x.

Значення цього об’єкту буде повернуто у місце виклику.

Інші методи та властивості defaultdict не відрізняються від методів та властивостей для стандартних словників (dict).

Приклад: кількість входжень слів у послідовність

Нехай з клавіатури вводиться послідовність слів (рядків).

Треба порахувати кількість входжень кожного слова.

OrderedDict

Клас OrderedDict призначено для створення та обробки словників, у яких зберігається порядок додавання елементів.

У звичайних словниках порядок повернення елементів (наприклад у циклі for … in …) є випадковим.

У деяких задачах зручно мати довідник, ключі якого повертаються у певному порядку.

Тоді варто застосувати OrderedDict.

OrderedDict.2

Для створення такого словника потрібно набрати

d = OrderedDict ()

Якщо до OrderedDict додається новий елемент, він потрапляє у кінець словника.

Якщо ж змінюється значення існуючого елемента, його позиція у довіднику не змінюється.

Дії над OrderedDict практично не відрізняються від дій над звичайним словником.

deque

Клас дек (deque) призначено для роботи з сукупностями елементів, до яких нам треба додавати або забирати елемент з початку або з кінця.

Створити дек можна за допомогою конструктора:

d = deque(t)

- де t належить типу, що ітерується

або

d = deque() – створює порожній дек.

Довжину деку можна обчислити за допомогою функції len:

len(d)

Основні методи роботи з деком

Метод	Опис
d.append(x)	Додати елемент x до кінця деку
d.appendleft(x)	Додати елемент x до початку деку
d.pop()	Забрати з деку та повернути останній елемент
d.popleft()	Забрати з деку та повернути перший елемент
d.rotate(n=1)	Перекласти n елементів з кінця до початку деку. Якщо n < 0, то перекласти n елементів з початку до кінця деку
d.clear()	Очистити дек

Приклад. Гра у відгадування слів

Реалізувати гру у відгадування слів, яка полягає у наступному.

По колу розташовані гравці (відгадувачі), яким презентують слово для відгадування.

Всі літери цього слова спочатку закриті (замінені зірочками, ‘*’).

Гравці вступають у гру по порядку. Кожен гравець може назвати літеру або слово.

Приклад. Гра у відгадування слів.2

Якщо гравець називає літеру, а цієї літери, у слові немає, - хід переходить до наступного гравця. Якщо ж така літера у слові є, то всі входження цієї літери у слово відкриваються, а гравцю нараховуються стільки балів, скільки є входжень названої літери у слово. Якщо всі літери слова відкриті, - гравець стає переможцем.

Якщо гравець називає слово і це слово не дорівнює заданому, то всі бали гравця анулюються, а хід переходить до наступного гравця. Якщо ж слово названо правильно, - гравець отримує стільки балів, скільки є у слові невідгаданих літер, та стає переможцем.

Переможець отримує премію: стільки балів, скільки літер було у слові.

Приклад. Гра у відгадування слів. Розв’язання

Для реалізації гри використаємо дек гравців (відгадувачів).

Опишемо клас Guesser (Відгадувач), у якому будемо зберігати ім’я гравця та кількість зароблених балів.

Слово будемо вибирати з текстового файлу наступним чином: знайдемо випадкове місце у файлі. Починаючи з цього місця, прочитаємо 10 рядків файлу, видалимо з них символи-розділювачі, переведемо до нижнього регістру та побудуємо список слів. З цього списку виберемо випадкове слово для відгадування.

Побудуємо також рядок, який буде містити закриті та вгадані літери вибраного слова (спочатку – всі зірочки).

Далі гравці будуть називати літери або слова а програма буде аналізувати відповіді та слідувати правилам гри до моменту, поки не буде відгадано задане слово.

Перевизначення операцій

Методи які починаються та закінчуються двома підкресленнями ‘__’, називають **особливими** або магічними.

Ми вже зустрічались з такими методами: конструктором __init__ та деструктором __del__.

Ми не викликаємо їх напряму. Натомість Python викликає ці методи у певних ситуаціях. Так, конструктор викликається під час створення об’єкту, а деструктор, - під час його знищення.

За допомогою особливих методів у Python можна перевизначити для власного класу практично всі стандартні операції.

Наприклад, щоб перевизначити для класу операцію ‘+’, треба описати у класі реалізацію метода __add__. Тоді для двох об’єктів цього класу x та y Python буде трактувати x + y як x.__add__(y).

Неповний перелік операцій для перевизначення та відповідних особливих методів наведено у таблицях нижче.

Перевизначення операцій. Бінарні операції

Операція	Метод
x + other	x.__add__(other)
x - other	x.__sub__(other)
x * other	x.__mul__(other)
x // other	x.__floordiv__(other)
x / other	x.__truediv__(other)
x % other	x.__mod__(other)
x ** other	x.__pow__(other)

Перевизначення операцій. Присвоєння спеціального виду

Операція	Метод
x += other	x.__iadd__(other)
x -= other	x.__isub__(other)
x *= other	x.__imul__(other)
x /= other	x.__idiv__(other)
x //= other	x.__ifloordiv__(other)
x %= other	x.__imod__(other)
x **= other	x.__ipow__(other)

Перевизначення операцій. Унарні операції

Операція	Метод
- x	x.__neg__()
+ x	x.__pos__()
len(x)	x.__len__()
abs(x)	x.__abs__()
complex(x)	x.__complex__()
int(x)	x.__int__()
long(x)	x.__long__()
float(x)	x.__float__()

Перевизначення операцій. Відношення

Операція	Метод
x < other	x.__lt__(other)
x <= other	x.__le__(other)
x == other	x.__eq__(other)
x != other	x.__ne__(other)
x >= other	x.__ge__(other)
x > other	x.__gt__(other)

Перевизначення операцій. Дії над послідовностями та словниками, виклик

Дія	Метод
x[key]	x.__getitem__(key)
x[key] = value	x.__setitem__(key, value)
del x[key]	x.__delitem__(key)
x(other)	x.__call__(other)

Приклад. Клас Polynome

Реалізувати клас Поліном. Передбачити методи для виконання арифметичних операцій над поліномами.

Опишемо клас Polynome як нащадок defaultdict.

У реалізації класу опишемо особливі методи для виконання арифметичних операцій (__add__, __radd__, __sub__, __rsub__, __mul__, __rmul__) а також методи __call__ для обчислення значення поліному та __str__ для отримання представлення поліному у вигляді рядка.

Окрім цього опишемо методи для обчислення похідної поліному (deriv), додавання одночлена до поліному (add_monom) та отримання степені поліному (get_degree).

Резюме

Ми розглянули:

1. Класи та атрибути
2. Класи та модулі
3. Наслідування та агрегування
4. Стандартні контейнери Counter, defaultdict, OrderedDict, deque
5. Перевизначення операцій

Де прочитати

1. Обвінцев О.В. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
3. Python 3.4.3 documentation
4. Hellmann D. - The Python Standard Library by Example – 2011
5. http://www.python-course.eu/python3_magic_methods.php
6. http://www.programiz.com/python-programming/operator-overloading