ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 13. Класи та об’єкти

Об’єктно-орієнтоване програмування

- Розглянуте раніше модульне програмування забезпечує високий рівень абстракції алгоритмів та даних.
- Але розвиток програмної індустрії виявив також обмеження, яким підпорядковані модулі.
- Зокрема, це необхідність часто багаторазово повторювати дуже схожі підпрограми або модулі для реалізації споріднених, але все ж різних, понять.
- Еволюція призвела до появи у середині 80-х років XX століття об’єктно-орієнтованого програмування.

Об’єктно-орієнтоване програмування.2

- Об’єктно-орієнтоване програмування спочатку з’явилось як альтернатива звичайному процедурному програмуванню.
- Однак подальший розвиток показав більшу ефективність поєднання можливостей процедурного та об’єктно-орієнтованого програмування.
- Таким чином, розповсюджені мови програмування, наприклад, C, набули нових об’єктно-орієнтованих можливостей.
- Взагалі, об’єктно-орієнтований підхід не обмежується тільки програмуванням.
- Він широко застосовується також при аналізі та проектуванні великих програмних систем.
- Але у даному курсі ми розглянемо саме об’єктно-орієнтоване програмування.

Об’єкти та класи

- **Об’єктом** називають деяку сутність, яка має визначені властивості, поведінку та стан.
- Множину об’єктів з однаковими властивостями та поведінкою називають **класом**.
- Об’єкти часом називають також екземплярами класу, хоча ці два поняття не є тотожними.
- Кожен клас має властивості та методи.
- **Властивості** визначають та зберігають значення властивостей об’єктів класу, а **методи** визначають поведінку об’єктів класу.

Наслідування

- Головне, що відрізняє об’єкти від розглянутих раніше структур даних, - це наявність наслідування.
- **Наслідування** – це успадкування деяких властивостей та методів одного класу іншим.
- Таким чином, з точки зору наслідування виділяються **клас-предок** та **клас-нащадок**.
- Клас предок також називають **суперкласом**, а клас-нащадок – **підкласом**.
- Наслідування дозволяє будувати ієрархію класів.
- Наслідування може бути одинарним або множинним.
- При **одинарному** наслідуванні клас наслідує властивості та методи тільки одного класу.
- При **множинному** наслідуванні клас може наслідувати властивості та методи більше, ніж одного класу.

Класи у Python

- Класи у Python позначаються ключовим словом class.
- Властивості класу у Python називають полями, методи – методами.
- Разом властивості та методи називають атрибутами класу.
- Клас містить описи методів та властивості.

Класи у Python.2

- Загальний синтаксис класу виглядає так:

```
class A:
    def f1(...):
        P1

    def f2(...):
        P2
...

    def fn(...):
        Pn
```

  - де A – ім'я класу, f_1, …, f_n – імена методів класу, P_1, …, P_n – реалізація методів класу.
- Кожний метод у списку параметрів повинен мати обов'язковий перший параметр, який прийнято називати self.
- Цей параметр використовують для позначення об'єкту для якого викликають метод.

Створення та використання об'єктів класу

- При створенні об'єктів класів використовують спеціальний метод __init__.
- Взагалі, взяття у Python імені з обох боків у подвійні підкреслення (__m__) означає, що це ім'я має спеціальне призначення.
- Так, Python викликає метод __init__ під час створення об'єктів класу.
- Для створення об'єкту класу A треба змінній, яка і буде цим об'єктом, присвоїти вираз

t = A(x_1, ..., x_n)

  - де x_1, ..., x_n – фактичні параметри для формальних параметрів, вказаних у методі __init__, за виключенням self.
- Подальше використання створеного об'єкту – це використання його полів та виклик методів.
- Для використання поля a_i об'єкту t треба вказати

t.a_i

- Для виклику метода f_j об'єкту t треба вказати

t.f_j(e_1, ..., e_k)

  - де e_1, …, e_k – фактичні параметри для формальних параметрів, вказаних у методі f_j, за виключенням self.

Поля класу

- Поля класу визначаються у методах класу присвоєннями вигляду

self.a_i = e

  - де a_i – ім’я поля, e – вираз.
- Як правило, початкові значення полів класу визначають під час створення об’єкту, тобто у методі __init__.

Наслідування

- Той факт, що клас *B* є нащадком класу *A*, позначають у описі класу *B*

```
class B(A):
    ...
```

- Python підтримує також множинне наслідування, з яким ми познайомимось пізніше.
- При множинному наслідуванні класи, від яких успадкований даний клас, вказують у дужках через кому.

Наслідування.2

- Клас-нащадок успадковує всі поля та методи класу предка окрім власних методів, які мають однакові імена з методами класу-предка.
- Кажуть, що останні методи класу-нащадка **перевизначають** відповідні методи класу-предка.
- Якщо у реалізації деякого методу класу-нащадка *B* необхідно викликати метод *f* класу-предка *A*, це позначають наступним чином:

***A.f*(self, ...)**

Приклад

- Скласти програму для обчислення нарахованої студентам стипендії в залежності від результатів сесії.
- Опишемо клас Person (особа) та його клас-нащадок Student (студент).

Зв’язування між об’єктами та методами

- Зв’язування між об’єктами та методами визначає, коли саме метод буде співставлений з об’єктом.
- Існує два типи зв’язування об’єктів та методів: статичне та динамічне.
- При **статичному** зв’язування метод для об’єкта визначається до початку виконання програми, тобто під час компіляції.
- При **динамічному** зв’язуванні метод призначається під час виконання програми.
- Методи, для яких застосовують динамічне зв’язування, називають **віртуальними**.
- Віртуальні методи дозволяють реалізувати **поліморфізм**, коли однакова поведінка реалізована для різних класів (часто - реалізована по-різному).
- Слід зазначити, що у Python всі методи є віртуальними.

Обмеження доступу до полів та методів класу

- Часто виникає необхідність обмежити доступ до деяких полів та методів класу, зробити їх внутрішніми у класі або у класі та його нащадках.
- Такі поля та методи у мовах програмування часто називають приватними та/або захищеними.
- На жаль, у Python немає можливості обмежити доступ на рівні мови.
- Але є стандартні домовленості щодо іменування таких внутрішніх полів та методів: їх імена повинні починатись з підкреслення ‘_’.
- Такі властивості та методи не рекомендовано використовувати ззовні класу.
- Треба також зазначити, що для полів та методів, імена яких починаються з двох підкреслень ‘__’, Python змінює ці імена, додаючи до початку ім’я класу.
- Звичайно, це не знімає небезпеку використання таких імен ззовні, але обмежує випадкове використання.

Рядки документації класу

- Для класів, як і для функцій та модулів, застосовують рядки документації, обмежені з обох боків трьома апострофами або подвійними лапками.
- Рядок документації класу повинен починатись у наступному рядку після імені класу на рівні опису методів та полів класу.
- Рядки документації методів класу оформлюють так само, як і рядки документації функцій.

Графічна бібліотека turtle

- Для наступного прикладу нам знадобиться графічна бібліотека turtle (черепаха).
- Цей модуль дозволяє зображувати прості фігури у графічному режимі.
- turtle – це графічний курсор зі спрямуванням, який можна пересувати по екрану.
- Модуль turtle містить багато різноманітних функцій. Розглянемо тільки деякі дії над turtle:

Графічна бібліотека turtle.2

Дія	Опис
turtle.home()	Перевести курсор у початкову позицію (центр вікна)
turtle.delay(pause)	Визначити затримку у мілісекундах між окремими рухами курсора
turtle.up()	Підняти пензель догори (припинити малювання)
turtle.down()	Опустити пензель донизу (почати малювання)
turtle.setpos(x, y)	Встановити курсор у позицію (x, y)
turtle.pencolor(c)	Встановити колір ліній за угодою у значення c
turtle.dot(c)	Зобразити точку на екрані. Якщо c не вказано, то поточним кольором переднього плану. Якщо c вказано, то кольором c.
turtle.circle(r)	Зобразити коло на екрані радіусом r поточним кольором переднього плану. Коло буде зображуватись, починаючи з його нижньої точки.
turtle.bgcolor()	Повернути поточний колір фону
turtle.pencolor()	Повернути поточний колір переднього плану
turtle.bye()	Завершити роботу turtle

Приклад

- Скласти модуль для зображення та переміщення точок та кіл по екрану (версія 1).
- Опишемо два класи: точка екрану (Point) та коло (Circle).

Статичні поля та статичні методи класів

- Окрім вже розглянутих полів та методів, визначених для об’єктів класу, існують також статичні поля та статичні методи класу.
- **Статичні поля** та **статичні методи** класу існують у одному примірнику, а не для кожного окремого об’єкту.
- Всі зміни, які деякий об’єкт класу здійснює над статичним полем, відобразяться на всіх об’єктах даного класу.
- Таким чином, статичні поля можуть використовуватись для обміну інформацією між декількома об’єктами одного класу.
- Статичні поля ініціалізують всередині опису класу як звичайні змінні.
- Статичні методи також належать класу, а не об’єкту.
- Такі методи застосовують як звичайні функції без прив’язки до конкретного об’єкту, але вказуючи клас, до якого належить метод.
- Статичні методи спочатку описують як звичайні методи класу, тільки не включають до списку параметрів self. Але після опису додають рядок

  f = staticmethod(f)

  - який вказує на те, що метод *f* є статичним методом класу.

Модуль для роботи з псевдовипадковими величинами random

- Модуль для роботи з псевдовипадковими величинами називається random та застосовується для генерації псевдовипадкових дійсних та цілих чисел а також для дій над послідовностями.

Модуль для роботи з псевдовипадковими величинами random.2

- Основні функції модуля:

Дія	Опис
random.seed(a=None)	Ініціалізувати генератор. Якщо a відсутнє або a == None, для ініціалізації використовують системний час. Якщо a – ціле, то використовують a.
random.randrange(start, stop[, step])	Повертає псевдовипадкове ціле число у діапазоні від start до stop-1 з урахуванням step.
random.choice(seq)	Вибирає з послідовності seq один псевдовипадковий елемент.
random.shuffle(seq)	Перемішує послідовність seq
random.sample(population, k)	Вибирає із population k випадкових елементів
random.random()	Повертає псевдовипадкове дійсне число у діапазоні [0,1).
random.uniform(a, b)	Повертає псевдовипадкове дійсне число у діапазоні від a до b.

Приклад

- Скласти модуль для зображення та переміщення точок та кіл по екрану (версія 2).
- Скласти програму для тестування цього модуля в якій випадковим чином на екрані зображуються та переміщуються точки та кола.
- У кінці роботи програма виводить кількість точок та кіл.

Загальне зауваження щодо класів та об’єктів у Python

- Тепер ми можемо повернутися трохи назад та розглянути раніше розібрані теми.
- У Python всі змінні є об’єктами. Навіть змінні числових типів.
- Рядки, списки, кортежі, словники також є об’єктами стандартних класів.
  - Це, зокрема, пояснює синтаксис виклику операцій та інструкцій для рядків, списків, кортежів, словників з використанням крапки – це, по суті, виклик методів для об’єктів класу.
- Більше того, функції та модулі також є об’єктами відповідних класів.
- І нарешті класи є об’єктами інших класів (метакласів).
- Тому, подібно Піфагору, який стверджував, що «Все є число», у Python ми сміливо можемо сказати: «Все є об’єкт».

Збереження змінних у пам’яті

- У темі «Числові типи даних» ми розглядали, яким чином дані зберігаються у пам’яті.
- Але тоді не було зрозуміло, чому, наприклад, для дійсних чисел під саме число виділяється 8 байт, а загальний об’єм пам’яті під відповідну змінну дійсного типу – 16 байт.
- Справа в тому, що Python окремо зберігає об’єкти та їх значення.
- Сам об’єкт є посиланням на його значення.
- Тобто, об’єкт (а будь-яка змінна є об’єктом) містить адресу пам’яті, у якій зберігається значення цього об’єкту.
- Декілька змінних можуть посилатися на одну адресу пам’яті, якщо вони мають однакове значення.
- Тому у Python змінні слід порівнювати не з комірками, у яких зберігається значення змінної, а радше з ярликами, які «навішують» на значення змінної.

Визначення типу об’єкту та відношення для об’єктів

- Щоб визначити тип об’єкту (а отже, - і будь-якої змінної), використовують стандартну функцію type.
- Якщо x – змінна, то type(x) повертає тип цієї змінної.
- Для об’єктів type(x) повертає клас, до якого належить об’єкт. Отже, ми можемо дізнатись, до якого типу належить той чи інший об’єкт.
- Ще одне відношення, яке визначено для об’єктів, - відношення is.
- Для двох змінних x, y відношення x is y буде істинним (True), якщо вони посилаються на один і той же об’єкт (значення об’єкту) у пам’яті.
- Наприклад, після

x = 2

y = 2

- відношення

x is y

- поверне значення True, оскільки Python містить один об’єкт (2) у пам’яті, а x, y посилаються на нього.

Визначення типу об’єкту та відношення для об’єктів.2

- Нарешті, стандартна функція isinstance(x, cls) перевіряє, чи належить об’єкт x класу cls або одному з його нащадків.
- Наприклад, після присвоєнь, вказаних вище, функція

isinstance(x,int)

- поверне значення True.

Резюме

- Ми розглянули:
  1. Об’єктно-орієнтований підхід до програмування.
  2. Визначення класів та об’єктів, інші поняття ООП
  3. Синтаксис опису класів у Python
  4. Створення та використання об’єктів класу.
  5. Зв’язування між об’єктами та методами. Віртуальні методи.
  6. Обмеження доступу до полів та методів класу
  7. Рядки документації класу
  8. Графічна бібліотека turtle
  9. Статичні поля та статичні методи класів
  10. Модуль для роботи з псевдовипадковими величинами
  11. Збереження змінних у пам’яті
  12. Визначення типу та відношення для об’єктів

Де прочитати

1. A Byte of Python (Russian) Версия 2.01 Swaroop C H (Translated by Vladimir Smolyar), http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf
2. Бублик В.В., Личман В.В., Обвінцев О.В.. Інформатика та програмування. Електронний конспект лекцій, 2003 р.,
3. Марк Лутц, Изучаем Python, 4-е издание, 2010, Символ-Плюс
4. Python 3.4.3 documentation
5. Марк Саммерфилд, Программирование на Python 3. Подробное руководство. - Символ-Плюс, 2009.
6. http://www.python-course.eu/python3_object_oriented_programming.php
7. http://foobarnbaz.com/2012/07/08/understanding-python-variables/