Programmazione Avanzata e parallela

Lezione 20

Luca Manzoni

Python e Oggetti

Programmazione orientata agli oggetti (OOP)

- Cosa è OOP?
- Creazione di una classe
- Costruttore
- Attributi
- Metodi

Python e Oggetti

Cosa è OOP

- Pensiamo a una lampadina
  - C’è il concetto di lampadina che ha alcune caratteristiche:
    - Cose che si possono fare (accendere/spegnere)
    - Stato interno (se è accesa/spenta)
- Una specifica lampadina rispetta il “template” della lampadina ma ha alcune caratteristiche fissate
- Due istanze del concetto di lampadina (i.e., due lampadine) avranno stato interno diverso ma possiamo svolgere le stesse operazioni su di loro

Python e Oggetti

Cosa è OOP

- La programmazione orientata agli oggetti è un modo di strutturare i programmi aggregando in “oggetti”:
    - Il comportamento
    - Le proprietà / stati
- Esempio: un oggetto può rappresentare una lampadina:
    - Proprietà: acceso/spento, tipo (led/incandescenza), Watt
    - Comportamento: “accendi”, “svita”, etc.

Python e Oggetti

Cosa è OOP

- Entità del mondo reale (o anche no) sono modellate come oggetti software, che sono modellati con:
  - Un insieme di dati associato a ogni entità
  - Un insieme di comportamenti per ciascuna entità
- In teoria possiamo, una volta modellata una entità, modellare altre entità in modo separato e rendere il software più mantenibile…
- …però serve introdurre il programmatore ai concetti di questo paradigma di programmazione

Python e Oggetti

Cosa è OOP: La classe “Cat”

- Si consideri un **classe** “Cat”:
  - Rappresenta il concetto generico di un gatto
  - Dobbiamo indicare che attributi ha (e.g., nome ed età). Questi sono gli **attributi** dell’istanza
  - Indichiamo che comportamenti può avere (e.g., miagolare). Questi sono i **metodi**
- Un gatto specifico può essere Tom, che avrà età==13 e nome==“Tom”
  - Essendo una istanza della classe “Cat” sappiamo che potrà miagolare

Python e Oggetti

Classi in Python

- In Python le classi sono un modo di definire strutture dati (e operazioni su di esse) da parte dell'utente
- Una classe fornisce una struttura ("blueprint") per qualcosa che deve essere definito, senza però fornire il contenuto (i.e., i valori degli attributi specifici)
- Per la classe "Cat" specificherà che un gatto deve:
  - Avere un nome e una età (ma non dirà che valori sono questi)
  - Che un gatto può miagolare

Python e Oggetti

Classi in Python

- Per definire una classe in Python usiamo:
  - **class NomeClasse:**
  - Tutta il codice contenuto all'interno sarà parte della classe…
  - …ma che codice dobbiamo mettere?
  - Se vogliamo solo dire che una classe esiste, senza dire nulla di attributi e metodi (proprietà e comportamenti) possiamo semplicemente usare la keyword "pass"
  - **class Cat:**
    pass

Python e Oggetti

Creazione di una istanza

- Così facendo abbiamo creato una classe “Cat”…
- …ma non abbiamo creato una istanza specifica di “Cat”
- La classe è un “blueprint” (i.e., una idea astratta)…
- …dobbiamo ora creare una istanza specifica di “Cat”
- Però per il momento non abbiamo definito nessun comportamento specifico o nessuna proprietà…
- …come possiamo fare in Python?

Python e Oggetti

Attributi di istanza

- Tutte le classi sono usate per creare oggetti e tutti gli oggetti contengono delle caratteristiche/proprietà/attributi

- Questi possono venire impostati al momento della creazione dell'oggetto (i.e., quando "riempiamo" il blueprint in caso concreto)

- Al momento della creazione viene chiamato il metodo __ **init(self, …)**__

  - Questo metodo è utilizzato per inizializzare gli attributi di un oggetto al momento della creazione

  - Questo metodo (o equivalente in altri linguaggi) è chiamato il costruttore

Python e Oggetti

Il metodo __init__

- Il metodo __init__ ha come argomenti:
  - “self”, che indica l’oggetto stesso (equivalente a “this” in linguaggi come Java)
    - Se fossimo in C “self” sarebbe un puntatore alla struttura stessa, in modo da poter dire “voglio accedere al campo x di questa struttura”
  - Gli argomenti dopo il primo possono essere aggiunti per inizializzare lo stato dell’oggetto (e.g., impostare gli attributi)

Python e Oggetti

Il metodo __init__

- Possiamo aggiungere due argomenti al costruttore di Cat, name e age.

- Per salvarli come attributi possiamo usare **self.nome_attributo = …**
  Se non era già presente viene creato sul momento

- Una volta creata l'oggetto e assegnato a una variabile possiamo accedere agli stessi attributi allo stesso modo anche fuori dal costruttore:

  - e.g.,
    x = Cat("Tom", 13)
    print(x.name)

Python e Oggetti

Attributi di classe

- In alcuni casi ci sono caratteristiche comuni a tutte le istanze di una classe
- In questo caso ha senso che siano definite a livello di classe e non di singola istanza
- Queste variabili sono definite dentro la classe ma fuori dal metodo __init__
- Saranno accessibili usando **NomeClasse.NomeAttributo**…
- …ma anche come se fossero normali attributi di una istanza

Python e Oggetti

Assegnamento: attributi di classe e di istanza

- Possiamo modificare gli attributi di una istanza senza problemi (chiaramente cambieranno solo per quella istanza)

- E per gli attributi di classe?
  - Se li modifichiamo accedendo tramite una istanza verranno modificati solo per quella istanza (che da quel momento in poi avrà una sua “copia locale”)

  - Se li modifichiamo accedendo tramite il nome della classe li modifichiamo per tutte le istanze

Python e Oggetti

Metodi di istanza

- Per definire il comportamento di tutte le istanze di una particolare classe possiamo definire dei metodi:
  - I metodi di istanza sono definiti, come il metodo __init__, all'interno di una classe
    - Il primo argomento è "self", ma possono avere quanti argomenti vogliamo
- Proviamo a implementare una classe "queue" (una coda)