Laboratorio di Programmazione

Andrea Mecchina Michele Rispoli Pietro Morichetti
Nicolas Solomita Stefano A. Russo (supervisione)

20 Novembre 2020 - Foglio 1 - versione 1.1

1 Un primo approccio a Python

Esercizio 01

Realizzare un programma con la seguente lista di funzioni per la manipolazione di liste:

- *stampa*, una funzione che stampa il contenuto di una lista passata come argomento;
- *statistiche*, una funzione che riceve una lista e, se è una lista di interi, ne determina la somma, la media, il minimo ed il massimo degli elementi;
- *somma_vettoriale*, una funzione che riceve in ingresso due liste, determina se sono due liste di interi, se hanno la stessa dimensione e ne calcola la somma vettoriale, poi ritornata come lista, altrimenti ritorna una lista vuota;
- *prodotto_vettoriale*, una funzione che riceve in ingresso due liste, determina se sono due liste di interi, se hanno la stessa dimensione, in caso ne calcola e ne ritorna il prodotto vettoriale, altrimenti ritorna una stringa di avviso "ATTENZIONE: non sono riuscito a calcolare il prodotto vettoriale".

Testare le funzioni passandogli in input diverse liste.

Nota Potete usare la funzione built-in *type* per verificare se una certa variabile è un (ogetto) int oppure no (es: type(var_int) == int -¿ true). Non serve importare alcun modulo.

2 Manipolazione dati su file

Esercizio 02

Implementare una funzione (simile a quella creata le lezioni precedenti) che ritorni una lista i cui elementi saranno le date delle vendite del file *shampoo_sales.csv*

Attenzione: Avevate usato la funzione *float* per convertire i prezzi da stringhe a valori numerici. Questa volta dovrete usare un'altra funzione:

```
from datetime import datetime
  ...
my_date = datetime.strptime(elements[0], '%d-%m-%Y')
```

Questa conversione vi permetterà di eseguire operazioni con le date (come trovare la data più lontana, quella più vicina, la conversione in mesi, ecc ecc..). Stampando la lista però, vederete degli oggetti di tipo DateTime. Se volete stampare le date in modo più leggibile, i comandi sono i seguenti:

```
for data in date_vendite:
    print(data.strftime('%d-%m-%Y'))
```

Esercizio 03

Estendere la classe CSVFile che avete creato la scorsa lezione, aggiungendo i seguenti metodi:

- *get_date_vendite()* : questa funzione ritornerà una lista con le date delle vendite. (Hint: usare la funzione creata al punto 2.1)
- *__str__()* : questa funzione ritornerà l'intestazione (header) del file CSV

3 Classi e Oggetti

Esercizio 04

Implementare una classe **Automobile** che presenta i seguenti attributi: casa_auto, modello, numero_posti, numero_portiere, kw, targa e categoria. Inoltre, la presente classe deve comprendere i seguenti metodi:

- *__init__*, metodo per inizializzare una istanza della classe;
- *__str__*, metodo che stampa tutte le informazioni associate ad una specifica istanza (aka oggetto) della classe Automobile;
- *parla*, metodo che stampa a schermo "Broom Broom";

- *confronta*, metodo che, dato in ingresso self e un'altra istanza di Automobile, determina se le due automobili hanno le stesse informazioni (eccetto per la targa che è univoca!).
- *bollo*, metodo che, dato in ingresso la categoria della vettura (Euro0, Euro1, ...), calcola il bollo dell'auto e lo ritorna. Si osservi che:
  - **Euro0** fino a $100Kw$ si pagano 3€ per Kw, mentre 4.50€ se superiore;
  - **Euro1** fino a $100Kw$ si pagano 2.50€ per Kw, mentre 4.35€ se superiore;
  - **Euro2** si pagano 3€ per Kw.

  Il bollo si calcola come $Kw * prezzo$.

Esercizio 05

Estendere l'esercizio 04 realizzando la sottoclasse **Transformer** della classe Automobile. La sottoclasse è caratterizzata dai seguenti attributi e metodi.

Attributi

- nome, ossia il nome dell'istanza della classe Transformer;
- generazione, ossia la generazione di appartenenza del Transformer come un intero positivo (1, 2, 3, ...);
- grado, ossia il grado militare (es. "soldato semplice", "sergente", "capitano", ...)
- fazione, ossia la fazione di appartenenza che deve essere "Autobot", oppure "Decepticon";
- reparto, ossia la divisione a cui fa parte l'istanza della classe quindi "corpo a corpo", "artiglieria leggera", "artiglieria pesante", "spionaggio", ... (insomma inventate voi);

Metodi

- sovrascrivere il metodo *__init__* per fare in modo che accetti i nuovi attributi;
- sovrascrivere il metodo *parla* che in base alla fazione di appartenenza dell'oggetto restituisce su schermo la frase "Noi siamo Autobots, proteggeremo ogni essere vivente" in caso faccia parte della fazione degli Autobots. Oppure la frase "Noi siamo Decepticons e l'AllSpark sarà nostro!", in caso l'oggetto faccia parte della fazione dei Decepticons.

- definire il metodo *scheda_militare* che stampa a schermo le informazioni "militari" di una istanza della classe Transformer.

Utilizzare *super()* dove lo si ritiene appropriato.

Opzionale (ma caldamente consigliato)

Considerate il seguente frammento di pseudo-codice che fa riferimento alle classi sviluppate negli esercizi 04 e 05.

```
Automobile auto_0 = Automobile(...);
...
Transformer t_0 = Transformer(...);
Transformer t_1 = Transformer(...);
...
```

Rispondere ai seguenti quesiti:

1. auto_0 è di tipo Automobile?
2. Identificare la superclasse e sottoclasse tra Transformer e Automobile.
3. t_0 è di tipo Transformer?
4. t_0 e t_1 sono dello stesso tipo?
5. auto_0 e t_1 sono dello stesso tipo?
6. t_1 potrebbe essere di tipo Automobile?
7. auto_0 potrebbe essere di tipo Transformer?

Potete verificare la vostra risposta usando le funzioni booleane built-in: *issubclass*(nome_super_classe, nome_sotto_classe), *type*(nome_variabile), *isinstance*(nome_variabile, nome_classe).