



Pattern Recognition
and Applications Lab

Organizzare il Codice

Docente: Ambra Demontis

Anno Accademico: 2023 - 2024

Corso di Laurea in Ingegneria Elettrica, Elettronica e Informatica



University of Cagliari,
Italy

Department of Electrical and
Electronic Engineering



Organizzare il Codice

Scrivere un intero programma in un singolo file .py non è una buona pratica di programmazione.

Per ottenere un codice ordinato **è importante dividere il codice in diversi file.**

Generalmente, ad esempio, ogni classe viene definita in un file ad-hoc.

I Moduli

In Python ogni file viene chiamato modulo.

Per utilizzare una classe o una funzione definita in un altro modulo, dobbiamo importarla.

Per importare una funzione o una classe si utilizza la sintassi:

```
from <nome_file> import <nome_funzione_o_classe>
```

NB: nome file è il nome del file senza contare l'estensione .py

I Moduli

Supponiamo di voler organizzare in file il codice visto in una lezione precedente che definiva la classe CLibro e creava un'istanza di quella classe...

Definizione di Classi - Definizione degli Attributi

Definiamo la classe CLibro

```
class CLibro:
```

```
    def __init__(self, titolo, autore, editore, prezzo, anno_publicazione):  
        self.titolo = titolo  
        self.autore = autore  
        self.editore = editore  
        self.prezzo = prezzo  
        self.anno_publicazione = anno_publicazione
```

Creazione di un'Istanza della Classe

Creiamo un'istanza della classe libro (oggetto appartenente alla classe libro).

```
titolo = input("inserisci il titolo del libro ")
autore = input("inserisci l'autore del libro ")
editore = input("inserisci l'editore del libro ")
prezzo = float(input("inserisci il prezzo del libro "))
anno_pubblicazione = int(input("inserisci l'anno di pubblicazione del libro "))

oggetto_libro = CLibro(titolo, autore, editore, prezzo, anno_pubblicazione)
```

I Moduli

Prima di tutto creeremo un modulo che conterrà la classe, i.e., il file chiamato “c_libro.py”. Questo file si presenterà così:

```
class CLibro:
```

```
...
```

Qui avremo tutto il codice del vecchio esempio che **definiva** la classe CLibro.

I Moduli

Creeremo poi un altro script, chiamato “main_creazione_libro.py”. Questo file conterrà il codice che creava un’istanza di classe CLibro. Per poter far questo dovremo prima importare la classe dall’altro modulo.

La prima istruzione di questo file sarà quindi:

```
from c_libro import CLibro
```

Avremo poi il vecchio codice che utilizzavamo per creare l’istanza.

I Moduli

Il file “main_creazione_libro.py” apparirà quindi così:

```
from c_libro import CLibro
```

```
titolo = input("inserisci il titolo del libro ")
```

```
autore = input("inserisci l'autore del libro ")
```

```
editore = input("inserisci l'editore del libro ")
```

```
prezzo = float(input("inserisci il prezzo del libro "))
```

```
anno_publicazione = int(input("inserisci l'anno di pubblicazione del libro "))
```

```
oggetto_libro = CLibro(titolo, autore, editore, prezzo, anno_publicazione)
```

```
oggetto_libro = print(oggetto_libro.calcola_prezzo_scontato(2020))
```

I Moduli

In Python è anche possibile importare tutto ciò che è contenuto in un file.
Per far questo dovremmo scrivere:

```
from c_libro import *
```

Dove l'asterisco sta per all (tutto).

Nota: è fortemente sconsigliato!

E' sconsigliato perchè non vi permette di avere controllo su cosa importate.
Potreste finire per importare una classe/funzione con lo stesso nome di una che avete definito ed ottenere comportamenti inattesi.

Utilizzo di Librerie

Possiamo utilizzare la stessa sintassi per importare delle funzioni delle librerie di Python. In quel caso la sintassi sarà:

```
from <nome_liberia> import <nome_funzione_o_classe>
```

Ad esempio, quando installiamo Python viene installata anche una libreria chiamata ***math***.

Le funzioni generalmente più utilizzate di questa libreria sono mostrate nella slide seguente.

La libreria Math

funzione	descrizione
<code>cos(x)</code>	coseno (x deve essere espresso in radianti)
<code>sin(x)</code>	seno (come sopra)
<code>tan(x)</code>	tangente (come sopra)
<code>acos(x)</code>	arco-coseno (x deve essere nell'intervallo $[-1, 1]$)
<code>asin(x)</code>	arco-seno (come sopra)
<code>atan(x)</code>	arco-tangente
<code>radians(x)</code>	converte in radianti un angolo espresso in gradi
<code>degrees(x)</code>	converte in gradi un angolo espresso in radianti
<code>exp(x)</code>	e^x
<code>log(x)</code>	$\ln x$
<code>log(x, b)</code>	$\log_b x$
<code>log10(x)</code>	$\log_{10} x$
<code>pow(x, y)</code>	x^y
<code>sqrt(x)</code>	\sqrt{x}

Utilizzo di Librerie

Supponiamo di voler utilizzare la funzione per il calcolo della radice quadrata.
Il codice sarà ad esempio:

```
from math import sqrt

numero = float(input("inserisci un numero"))
radice = sqrt(numero)
print("La radice è ", radice)
```

Packages

Quando un progetto cresce di dimensioni, organizzarlo in moduli (file) differenti non è più sufficiente. Diventa necessario organizzare i moduli.

In Python un **package** è una **cartella che organizza diversi moduli**.

Per far sì che una cartella venga riconosciuta da Python come package e che quindi sia possibile importare i moduli che contiene nella cartella è necessario creare un file chiamato: `__init__.py`

Packages - Import

Supponete di avere la seguente struttura di files, dove i nomi in verde sono cartelle.

progetto_ecommerce

main.py

ecommerce

__init__.py

database.py

prodotti.py

Supponete di voler importare nello script main.py che si trova fuori dal package ecommerce la classe CProdotto che è definita nel file prodotti.py.

Packages - Import Assoluti

Possiamo utilizzare la seguente sintassi:

```
from <percorso> import <nome_funzione_o_classe>
```

Dove *percorso* è la gerarchia di package che contengono il file dal quale vogliamo importare qualcosa.

Il percorso va scritto separando ogni cartella o file con un punto.

Packages - Import Assoluti

progetto_ecommerce

main.py

ecommerce

__init__.py

database.py

prodotti.py

Nel file main.py quindi avremo il seguente import:

`from ecommerce.prodotti import CProdotto.`

Questo tipo di import viene chiamato **import assoluto** perchè è specificato tutto il percorso dalla cartella del progetto fino al file.

Packages - Import Relativi

Per accedere ad altri moduli all'interno dello stesso package si possono utilizzare anche **import relativi**, cioè import nei quali non specifichiamo tutto il percorso.

Per riferirsi ad un **modulo che si trova esattamente nello stesso package** si può utilizzare questa sintassi:

```
from .<nome_file> import <nome_funzione_o_classe>
```

Packages - Import Assoluti

Supponiamo di voler accedere alla classe CProdotto, definita nel file prodotti.py dallo script database.py, che si trova nello stesso package.

```
progetto_ecommerce  
    main.py  
    ecommerce  
        __init__.py  
        database.py  
        prodotti.py
```

Nel file database.py potremmo avere il seguente import:

```
from .prodotti import CProdotto
```

Packages - Import Relativi

Per riferirsi ad un **modulo che si trova nel package “padre” di quello nel quale si trova lo script** dal quale vogliamo importare una classe o una funzione:

```
from ..<nome_file> import <nome_funzione_o_classe>
```

Packages - Import Assoluti

Supponiamo di avere la seguente struttura di package e moduli:

progetto_ecommerce

main.py

ecommerce

__init__.py

database.py

prodotti.py

pagamenti

__init__.py

pagamenti.py

Supponiamo di voler importare la classe CProdotti che si trova in prodotti.py nello script pagamenti.

Nb: prodotti.py si trova nel package ecommerce che è il package “padre” del package pagamenti.

Packages - Import Assoluti

Supponiamo di avere la seguente struttura di package e moduli:

progetto_ecommerce

main.py

ecommerce

__init__.py

database.py

prodotti.py

pagamenti

__init__.py

pagamenti.py

Supponiamo di voler importare la classe CProdotti che si trova in prodotti.py nello script pagamenti.

Nb: prodotti.py si trova nel package ecommerce che è il package “padre” del package pagamenti.

L’istruzione input che dovremmo usare è la seguente:

```
from ..prodotti import CProdotto
```

Note

Esistono diversi modi per effettuare gli import in modo che il progetto funzioni correttamente.

Tuttavia, per progetti semplici come quelli che realizzare questo corso, conviene utilizzare:

- **Import relativi**, quando il modulo si trova dentro un package e si vuole che quel modulo possa **essere importato da altri moduli ma non eseguito direttamente**.
- **Import assoluti**, quando il modulo deve **essere eseguito direttamente e non deve essere importato da altri moduli**.

Esercizio

Modificare come segue il progetto dato:

- 1) Nello script `c_lista_libri` dentro il package `anagrafica_libri` utilizzare un import relativo per importare la classe `CLibro` definita nello script `c_libro`.
- 2) Nello script `main` inserire due import assoluti:
 - a) uno per importare la classe `CListaLibri`, definita nello script `c_lista_libri` nel package `anagrafica_libri`
 - b) Uno per importare la classe `CAzienda`, definita nello script `c_azienda` nel package `anagrafica_clienti`

Provare poi ad eseguire lo script `main.py`

Soluzione

Modificare come segue il progetto dato:

- 1) Nello script `c_lista_libri` dentro il package `anagrafica_libri` utilizzare un import relativo per importare la classe `CLibro` definita nello script `c_libro`.

Soluzione:

```
from .c_libro import CLibro
```

Esercizio

Modificare come segue il progetto dato:

2) Nello script main inserire due import assoluti:

- a) uno per importare la classe CListaLibri, definita nello script c_lista_libri nel package anagrafica_libri
- b) Uno per importare la classe CAzienda, definita nello script c_azienza nel package anagrafica_clienti

Soluzione:

```
from anagrafica_libri.c_lista_libri import CListaLibri  
from anagrafica_clienti.c_azienza import CAzienda
```