

Programmare nel 2021

Fabio Pavesi

- Carrellata di argomenti di potenziale esplorazione futura
 - Per nulla approfondita
 - Software visto come strumento di supporto alla ricerca scientifica

- Programmazione
- Architettura software
- Deployment (dove far girare i propri servizi)

- Paradigmi di programmazione
- Linguaggi rilevanti nel mondo della ricerca
- Linguaggi adatti all'ingegnerizzazione di software strumento (o oggetto) di disseminazione

Programming language

A programming language is a notation for writing programs, which are specifications of a computation or algorithm – Wikipedia

Ha l'obiettivo di strutturare il codice in modo da renderlo più **leggibile** e **manutenibile**.

Lo strumento principale per rendere il codice leggibile e manutenibile è l'**incapsulamento**, sotto forma di

- Funzioni
- Moduli
- Package
- Librerie

Esempio di programmazione non strutturata

```
A = {'x': 0, 'y': 0}
B = {'x': 3, 'y': 8}
C = {'x': 12, 'y': -5}
```

```
AB = sqrt((A['x'] - B['x']) ** 2 + (A['y'] - B['y']) ** 2)
BC = sqrt((C['x'] - B['x']) ** 2 + (C['y'] - B['y']) ** 2)
```

```
ABC = AB + BC
```

Esempio di programmazione strutturata

```
def distance(a, b):  
    return sqrt((a['x'] - b['x']) ** 2 + (a['y'] - b['y']) ** 2)
```

```
A = {'x': 0, 'y': 0}
```

```
B = {'x': 3, 'y': 8}
```

```
C = {'x': 12, 'y': -5}
```

```
AB = distance(A, B)
```

```
BC = distance(B, C)
```

```
ABC = AB + BC
```


Esempio di programmazione ancora più strutturata

```
def distance(a, b):  
    return sqrt((a['x'] - b['x']) ** 2 + (a['y'] - b['y']) ** 2)  
  
def length(points):  
    previous_point = None  
    total_length = 0  
    for point in points:  
        if previous_point is not None:  
            total_length = total_length + distance(point, previous_point)  
        previous_point = point  
    return total_length  
  
A = {'x': 0, 'y': 0}  
B = {'x': 3, 'y': 8}  
C = {'x': 12, 'y': -5}  
  
ABC = length([A, B, C])
```

Moduli

Raccolgono le funzioni in file

Package

Raccolgono moduli in “directory”

Librerie

Raccolgono package in oggetti “trasportabili”

Si spinge oltre la programmazione strutturata, offrendo un incapsulamento ancora maggiore.

- **Object/Class:** A tight coupling or association of data structures with the methods or functions that act on the data. This is called a class, or object (an object is created based on a class). Each object serves a separate function. It is defined by its properties, what it is and what it can do. An object can be part of a class, which is a set of objects that are similar.
- **Information hiding:** The ability to protect some components of the object from external entities. This is realized by language keywords to enable a variable to be declared as private or protected to the owning class.

- **Inheritance:** The ability for a class to extend or override functionality of another class. The so-called subclass has a whole section that is derived (inherited) from the superclass and then it has its own set of functions and data.
- **Interface** (object-oriented programming): The ability to defer the implementation of a method. The ability to define the functions or methods signatures without implementing them.
- **Polymorphism** (specifically, Subtyping): The ability to replace an object with its subobjects. The ability of an object-variable to contain, not only that object, but also all of its subobjects.

Functional programming

In computer science, functional programming is a programming paradigm where programs are constructed by applying and composing functions. It is a declarative programming paradigm in which function definitions are trees of expressions that map values to other values, rather than a sequence of imperative statements which update the running state of the program. – Wikipedia

Machine learning

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence – Wikipedia

Tipicamente i dati da cui apprendere vengono divisi in

- training data
- test data

Gli approcci sono diversi, ma la principale divisione è tra **supervised** e **unsupervised** learning.

Della filosofia **unsupervised** fa parte il **reinforcement learning**.

Linguaggi rilevanti nel mondo della ricerca - 1/2

Python

- Sintassi semplice ed intuitiva per chi non programma di mestiere
- Adatto per
 - Programmazione strutturata
 - Programmazione ad oggetti (con caveat)
- Desktop o server

Javascript/Typescript

- Sintassi più strutturata, ispirata al C
- Adatto per
 - Programmazione strutturata
 - Programmazione ad oggetti (pur essendo, di fatto, un linguaggio “a prototipi”)
 - Programmazione funzionale
- Desktop, server e browser

HTML

- Non è un linguaggio di programmazione ma di *markup*
- Necessario per scrivere pagine Web

PHP

- Linguaggio nato per generare HTML dinamicamente
- Adatto per
 - Programmazione strutturata
 - Programmazione ad oggetti
- Server-only

Linguaggi adatti all'ingegnerizzazione di software strumento (o oggetto) di disseminazione - 1/2

C++

- È il linguaggio più vicino alla macchina
 - altissime prestazioni
- Adatto per
 - Programmazione strutturata
 - Programmazione ad oggetti
 - Programmazione funzionale
- Desktop, server e, compilando in WASM, browser

Linguaggi adatti all'ingegnerizzazione di software strumento (o oggetto) di disseminazione - 2/2

Java

- Linguaggio ad oggetti puro
 - molto elegante
- Adatto per
 - Programmazione ad oggetti
 - Programmazione funzionale
- Desktop e server

NodeJS

- Javascript lato server
- Adatto per
 - Programmazione strutturata
 - Programmazione ad oggetti (pur essendo, di fatto, un linguaggio "a prototipi")
 - Programmazione funzionale
- Desktop e server

Client-server

- Client-server
 - frontend
 - desktop
 - browser
 - backend

Web API

- REST - Representational state transfer
 - XML - **eXtensible Markup Language**
 - validazione automatica
 - pessima usabilità in Javascript
 - JSON - **JavaScript Object Notation**
 - ottima usabilità in Javascript

Web Clients

- HTML
- Javascript / Typescript
- **S**erver **S**ide **R**endering
- **S**ingle **P**age **A**pplication

Microservices

- Evoluzione del modello Client-Server
- Usa tipicamente REST (ma anche altre tecnologie)
- Ben supportati dall'approccio **D**omain **D**riven **D**esign

Dove far girare i propri servizi

Un po' di storia

- Server fisico
- **V**irtual **M**achine
- Container

QA Time

- Le slide sono disponibili su GitHub:
https://github.com/fabiopavesi/carrellata_informatica
- Contatti: fabio.pavesi at ingv.it