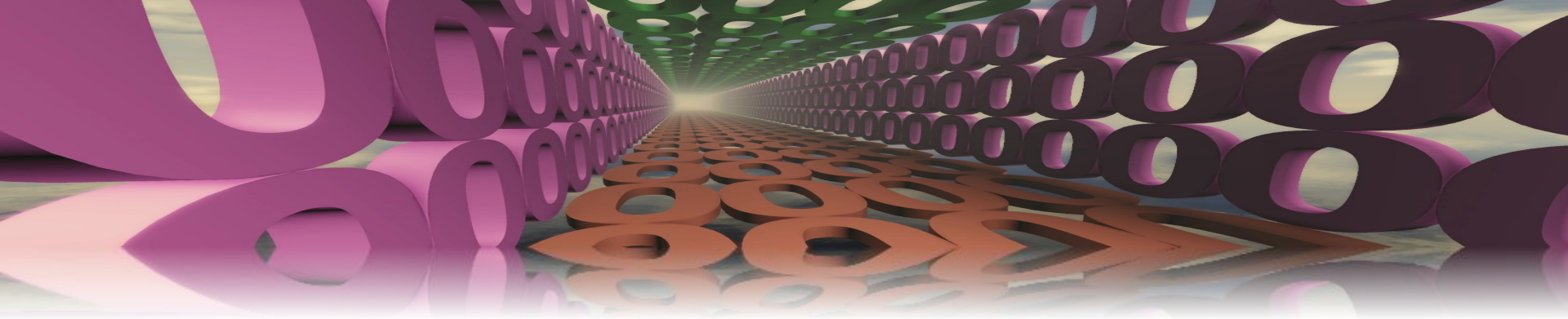


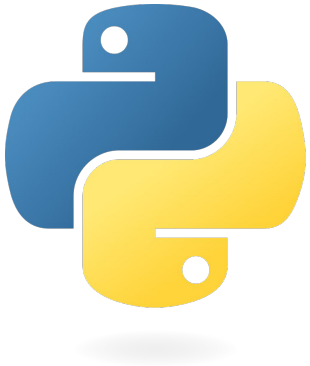
Level 0

Il prequel di CyberChallenge

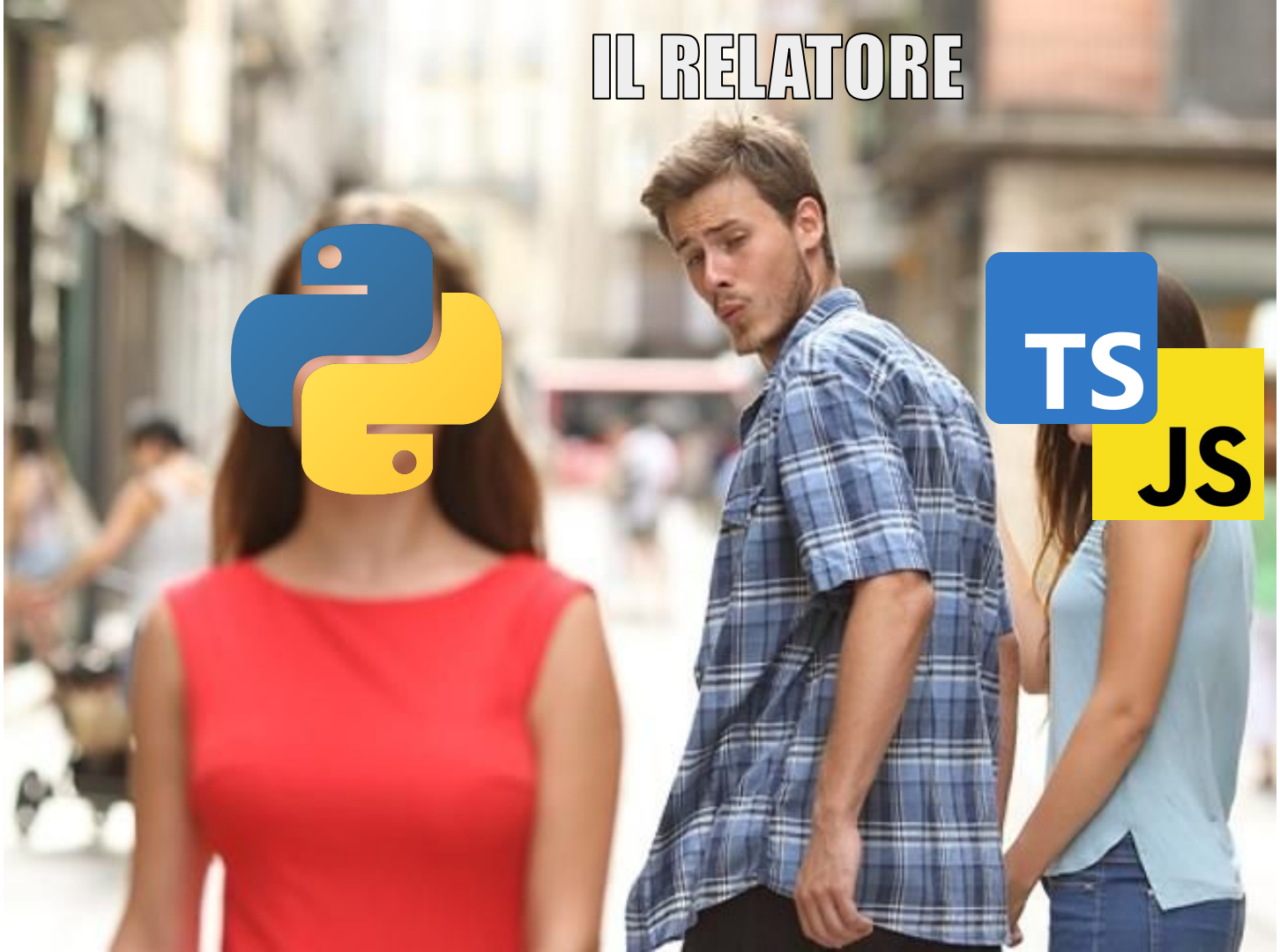
Samuele Casarin
samuele.casarin@unive.it

Agenda

- **Linguaggio Python**
  - Intro a Python
  - Crash Course: sintassi del linguaggio, strutture dati, gestione di file, RegEx
- **Encoding & Hashing**
  - Encoding/Decoding
  - Hashing e Funzioni hash crittografiche
- **Web Geeking**
  - Tecnologie Web
  - Chrome DevTools
  - Libreria *requests*



Linguaggio Python


IL RELATORE
TS
JS

Linguaggio Python

- Linguaggio ad alto livello
- Interpretato
- Dinamico
- Facile da imparare
- Python 2 ⫤ **Python 3**

```
# Python 2
print "Hello, world!"
```

```
# Python 3
print('Hello, world!')
```

Applicazioni

General-Purpose

- Web Development
- Data Science & Machine Learning
- Automation & Scripting
- Web Scraping
- Game Development
- Desktop Applications
- …

Cyber Security

- Vulnerability Scanning
- Penetration Testing
- Application Security Testing
- Web App Security Testing
- …

Perché Python?

- Semplice (e niente ‘;’)

```cpp
// C++
#include <iostream>
int main(char* args[]) {
  std::cout << "Hello, world!"
            << std::endl;
  return 0;
}
```

```python
# Python
print('Hello, world!')
```

Perché Python?

- Codice minimale e ordinato per costruzione (e niente ‘{’ o ‘}’ per blocchi)

```cpp
// C++
if (something) {
  alice;
} else if (something_else) {
  bob;
} else {
  charlie;
}
```

```python
# Python
if something:
  alice
elif something_else:
  bob
else:
  charlie
```

Perché Python?

- Funzionale

```cpp
// C++
bool areEqual(const std::vector<int>& a,
              const std::vector<int>& b) {
    return a.size() == b.size() &&
      std::equal(a.begin(), a.end(), b.begin());
}
int main() {
  std::vector<int> v1 = {1, 2, 3, 4};
  std::vector<int> v2 = {1, 2, 3, 4};
  std::cout << std::boolalpha << areEqual(v1, v2)
            << "\n"; // prints "true"
}
```

```python
# Python
[1, 2, 3, 4] == [1, 2, 3, 4]
→ True
```

Perché Python?

- Tipi dinamici (possibile supporto per annotazione di tipi!)

```cpp
// C++
int a = 8;
double b = 42.0;
std::string s = "Hello, world!";
auto z = 12;
```

```python
# Python
a = 8
# just numbers — no distinction
between int and float
b = 42
# same variable, different types
s = 15
s = 'Hello, world!'
z = 12
```

Perché Python?

- Operazioni senza sorprese

```
// JavaScript
("b" + "a" + + "a" + "a")
→ "baNaNa"
```

```
# Python
'Hours spent on debugging my
Node.js app: ' + 42
→ TypeError: can only concatenate
str (not "int") to str
```

Perché Python?

- Operazioni senza sorprese (come concatenare una stringa e un numero?)

```
# Python
'Hours spent on debugging my
Node.js app: ' + str(42)
→ 'Hours spent on debugging my
Node.js app: 42'
```

```
# Python
f'No. of hours spent debugging my
Node.js app: {42}'
→ 'Hours spent on debugging my
Node.js app: 42'
```

Perché Python?

- Un linguaggio *must-know* per la cybersecurity nel 2025
- Ecosistema vibrante e supporto attivo della comunità
- Gestione delle dipendenze con `pip` (Package Installer for Python)

```
$ pip install requests numpy pandas matplotlib flask django
scikit-learn tensorflow pytorch pycryptodome scapy hashlib paramiko
beautifulsoup4 selenium scrapy playwright socket httpx pygame panda3d
logging argparse …
```

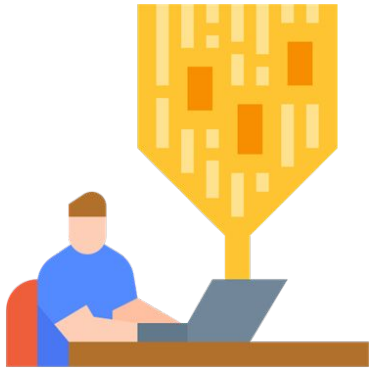
IT’S CRASH COURSE TIME!

```
$ apt install python3
```

Python: print

C++: cout

Java:
SysTEM.oUt.
prInTLn



Encoding & Hashing

Wuzzat?

- Encoding ≠ Encryption ≠ Hashing
- **Encoding/Decoding**: associare informazione a una stringa e viceversa
- **Encryption/Decryption**: associare una stringa (*plaintext*, testo in chiaro) a un'altra stringa (*cyphertext*, testo cifrato), in modo tale che dalla seconda NON sia possibile risalire facilmente alla prima, se non per mezzo di un segreto (*key*, chiave) *[ci vediamo alla lezione di crittografia…]*
- **Hashing**: associare una stringa di lunghezza arbitraria a una stringa di lunghezza fissa

Encoding/Decoding

- **Encoding/Decoding**: associare informazione a una stringa e viceversa

ASCII	Caratteri, es. Hello, world!
UTF-8	Caratteri, ma 与类固醇 (con steroidi) 👌
Base64	Dati, es. `SGVsbG8sIHdvcmxkIQ==`
URL encoding o Percent encoding	Dati, es. `Hello%2C%20world!`
JSON (usa UTF-8)	Numero, Stringa, Boolean, Lista e Dizionario
PNG	Immagine
WAV	Audio

ASCII

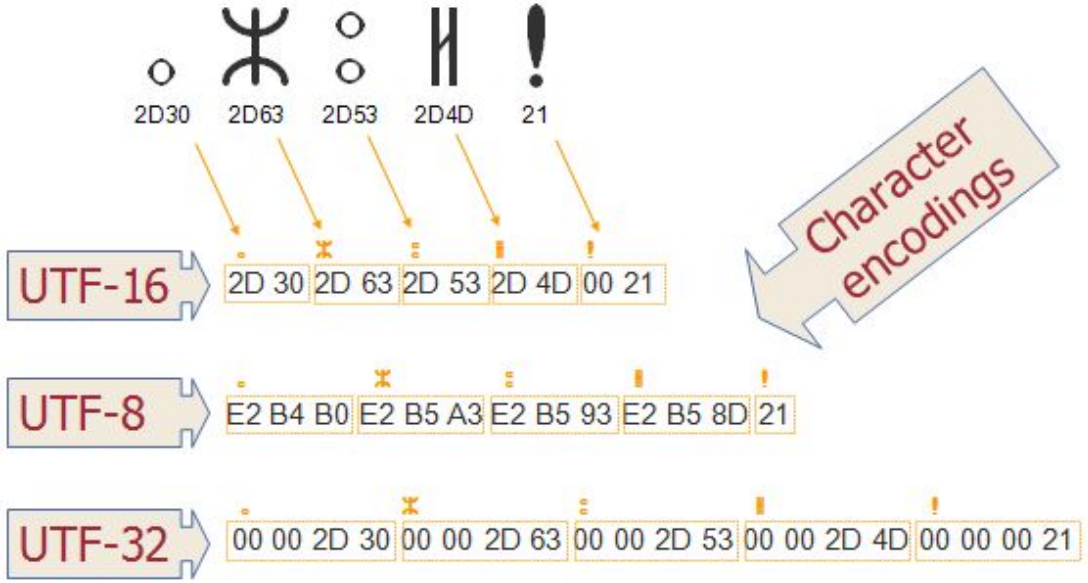
- Il formato più semplice di encoding/decoding di **caratteri**
- Ogni carattere è associato a un numero tra 0-127 (7 bit)
- Caratteri stampabili (32-126) e non stampabili (0-31,127)

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

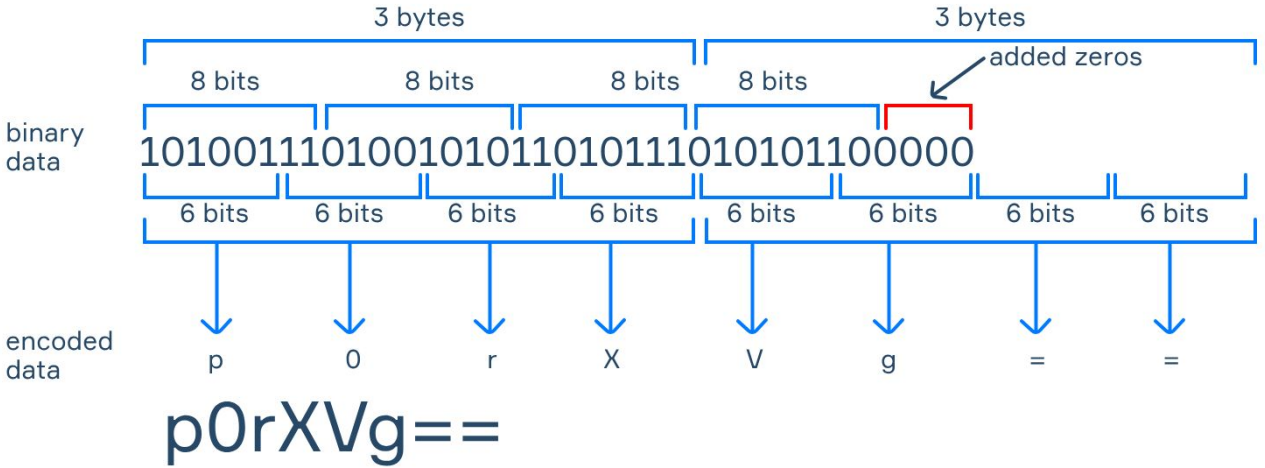
UTF-8

- Implementazione di Unicode (caratteri cinesi, giapponesi, greci, emoji, …)
- Ogni carattere è associato a una sequenza tra 1 e 4 byte
- UTF-16: Principale variante di UTF-8, sequenze di 2 o 4 byte



Base64

- Usato per rappresentare dati arbitrari con soli caratteri ASCII stampabili
- Divide una sequenza di byte in gruppi di 6 bit
- Associa a ogni gruppo (2^6=64 comb.) un carattere ASCII stampabile
- Se avanzano bit? Tra uno e due caratteri ‘=’ di padding



URL encoding o Percent encoding

- Usato per rappresentare dati arbitrari da inserire su URL
- Lascia invariati i byte associati a caratteri alfanumerici
- Converte ogni altro byte in ‘%xx’, dove ‘xx’ è il numero esadecimale (hex) associato al byte

JSON (JavaScript Object Notation)

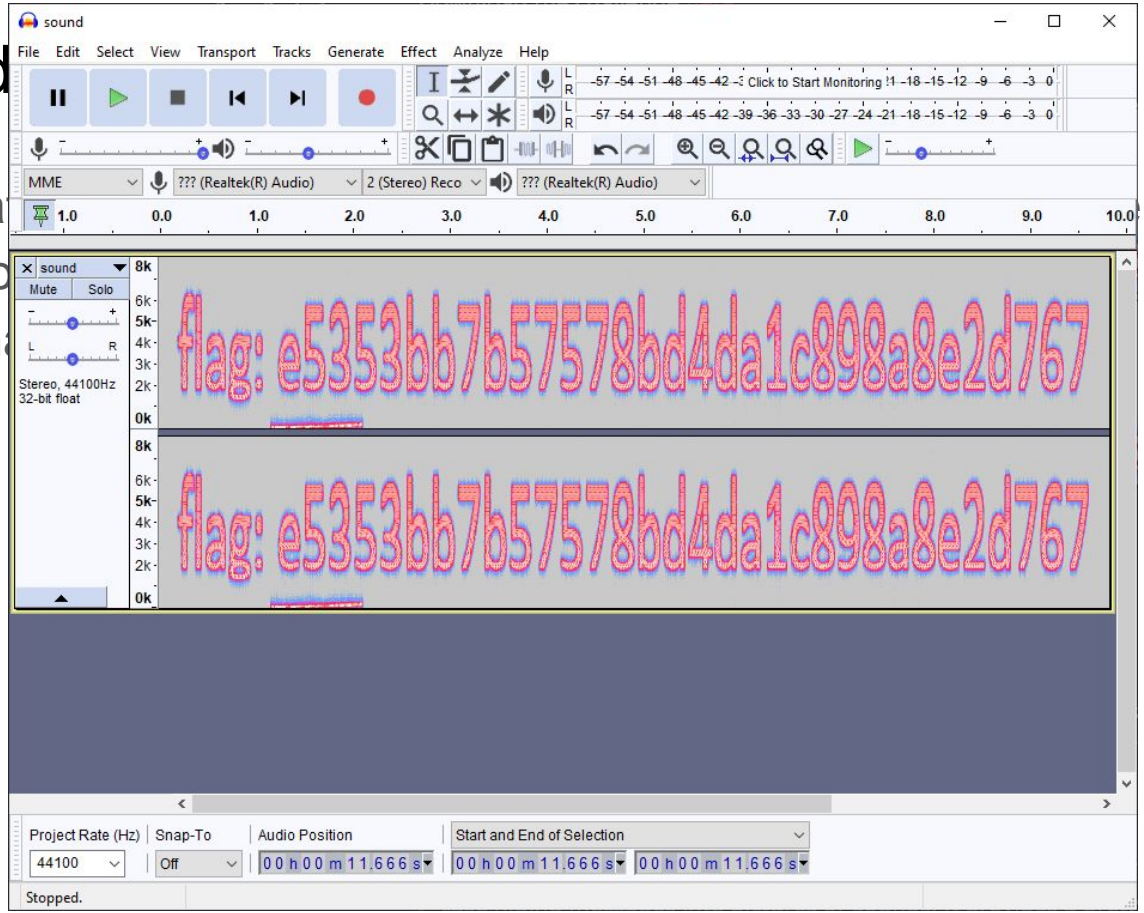
```
{
  "name": "Samuele Casarin",
  "age": 28,
  "isDeveloper": true,
  "programmingLanguages": ["JavaScript", "TypeScript", "Python", "C",
"C++", "Java", "PHP"],
  "canINestThings": {
    "wowDoItAgain": null
  }
}
```

Altri encoding

- Esiste praticamente uno schema di encoding per rappresentare ogni tipo di informazione in qualunque formato
- Alcune challenge nascondono flag in formati non convenzionali

Altri encod

- Esiste pra ogni tipo di informazio
- Alcune ch

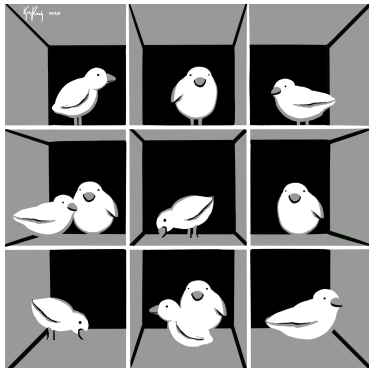


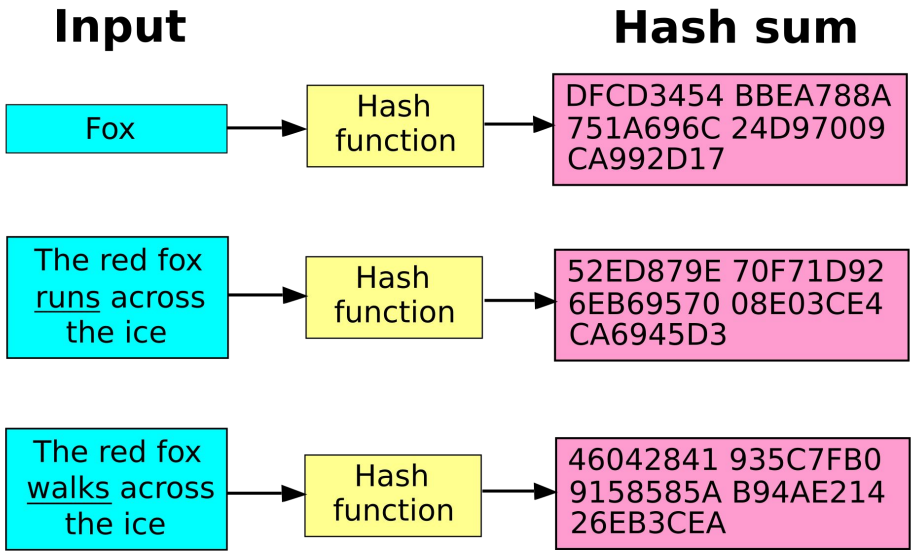
Encoding/Decoding in Python

```
'Hello, world!'.encode() # default: 'utf-8'
b'Hello, world!' # same as above

import base64
base64.b64encode('Hello, world!'.encode()).decode()
base64.b64decode('SGVsbG8sIHdvcmxkIQ=='.encode()).decode()

import json
data = json.loads('{"hello":"world"}')
json.dumps(data)
```

Hashing

- **Hashing**: associare una stringa di lunghezza arbitraria a una stringa di lunghezza fissa (solitamente corta), chiamata *digest*
- **Funzione hash**
- *One-way*, non invertibile
- Possibili collisioni





Funzioni hash crittografiche

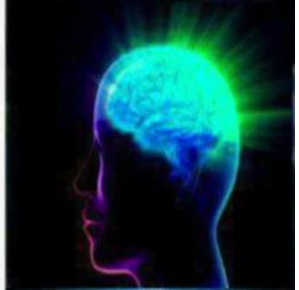
- Funzioni hash *f* che godono di proprietà crittografiche
  - È intrattabile, dato *h*, cercare *x* tale che *f(x)=h*
  - È intrattabile, dato *y*, cercare *x* tale che *f(x)=f(y)*
  - È intrattabile cercare *x* e *y* tali che *f(x)=f(y)* **COLLISIONE!**
- Funzioni hash crittografiche popolari: MD5, SHA128
- Funzioni hash crittografiche popolari e **sicure**: SHA256, SHA512
- Applicazioni
  - Controllo di integrità
  - Archiviazione sicura di password
  - Proof-of-Work (es. blockchain)
  - Identificatore di dati
  - Generazione di stringhe pseudocasuali e Derivazione di chiavi e password

Hashing in Python

```python
import hashlib

text = "Hello, world!"

md5_hash = hashlib.md5(text.encode()).hexdigest()
# 6cd3556deb0da54bca060b4c39479839

sha256_hash = hashlib.sha256(text.encode()).hexdigest()
# 315f5bdb76d078c43b8ac0064e4a0164612b1fce77c869345bfc94c75894edd3
```

CRC32

MD5



SHA-256

Doctor's prescription note

Web Geeking

Tecnologie Web

- HTTP e HTTPS
- Web Browser
- HTML
- JavaScript
- Cookie

***NB**:* “geeking” *perché non* è hacking
[ci vediamo alla lezione di web security…]

HTTP e HTTPS

HTTP: Hyper-Text Transfer Protocol

Protocollo (=insieme di regole, linguaggio) per il trasferimento di ipertesti

- Modello client-server
- Il client invia una **richiesta** al server
- Il server invia una **risposta** al client

HTTPS: HTTP Secure 🔒

Come HTTP, ma tutta la comunicazione è cifrata

(1) User issues URL from a browser
http://host:port/path/file

(5) Browser formats the response and displays

Client (Browser)

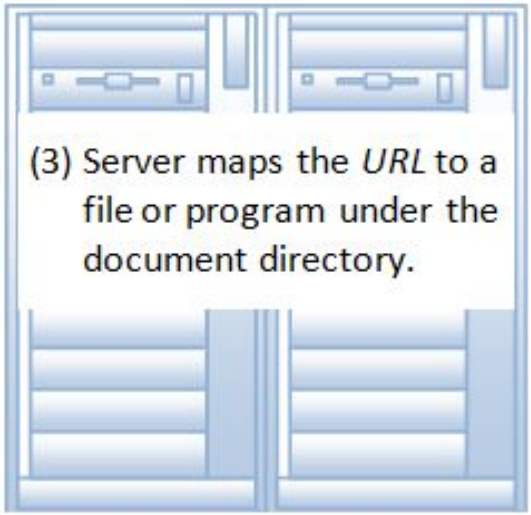
(2) Browser sends a request message

```
GET URL HTTP/1.1
Host: host:port
.................
.................
```

(4) Server returns a response message

```
HTTP/1.1 200 OK
.................
.................
.................
```

HTTP (Over TCP/IP)



(3) Server maps the *URL* to a file or program under the document directory.

Server (@ *host:port*)

Web Browser

Il software che consente la navigazione del Web

- Client HTTP(S)
- Interprete di diversi linguaggi
  - tra cui HTML e JavaScript
- Presentatore di pagine web

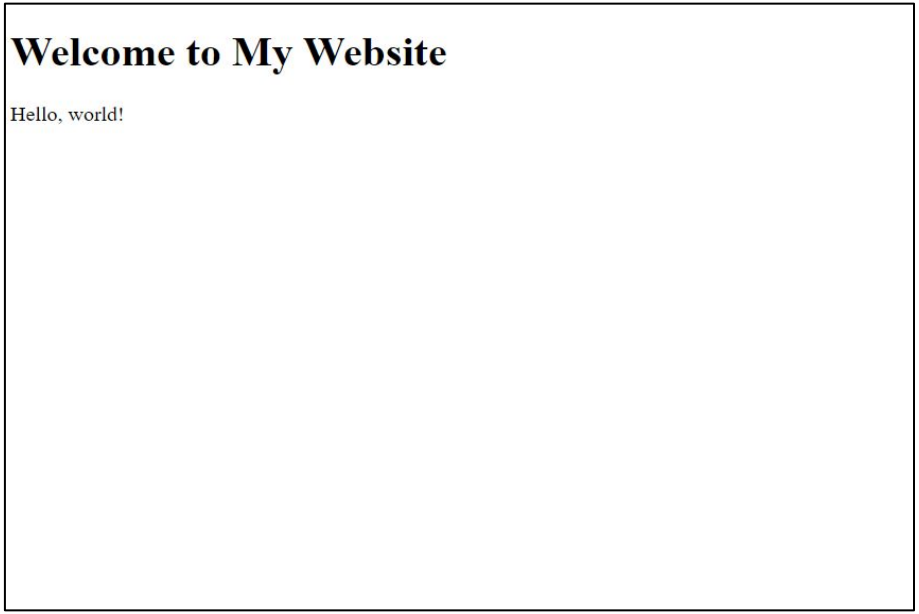
Esempi: Google Chrome, Mozilla Firefox

HTML (Hyper-Text Markup Language)

Linguaggio di markup (=formattazione) per ipertesti

- Il linguaggio standard per definire la **struttura** di pagine web
- Struttura ad **albero**
- I nodi dell'albero, chiamati **tag**, rappresentano elementi della pagina
- Gli elementi possono essere specializzati per mezzo di **attributi**

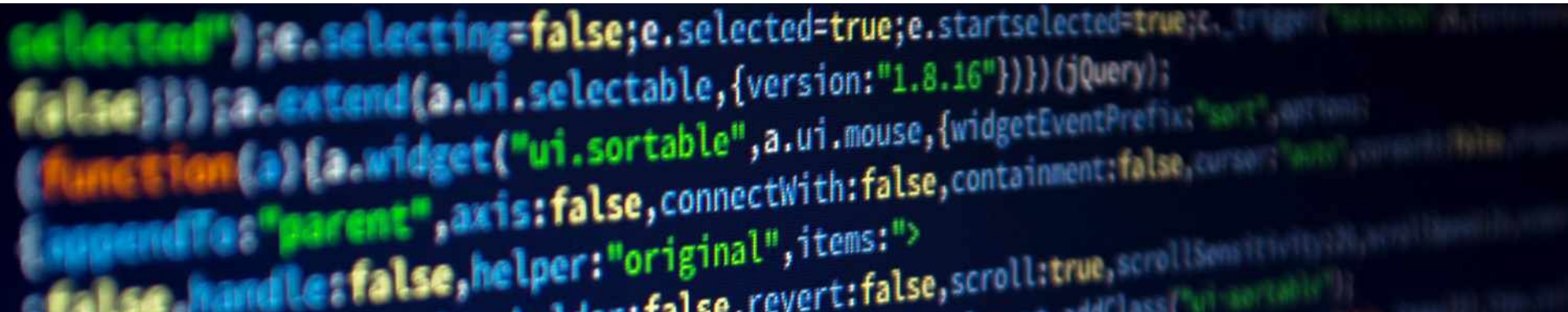
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1.0">
  <title>Simple HTML Document</title>
</head>
<body>
  <header>
    <h1>Welcome to My Website</h1>
  </header>
  <div>
    <p id="greeting">Hello, world!</p>
  </div>
</body>
</html>
```



JavaScript

Il linguaggio di scripting del web dal 1995

- **1995**: Gestire animazioni nelle pagine web
- **2025**: Sviluppare intere applicazioni web, desktop, mobile, …
- Gli script sono inclusi nelle pagine web attraverso il tag HTML `<script>`



```
<script>
  "use strict";
  (function () {
    var COLORS = ["#E81416", "#FFA500", "#FAEB36", "#79C314", "#487DE7"];
    var currentColorIndex = 0;
    setInterval(function () {
      document.getElementById("greeting")
        .style.color = COLORS[currentColorIndex];
      currentColorIndex = (currentColorIndex + 1) % COLORS.length;
    }, 500);
  })();
</script>
```

Cookie

Dati in forma chiave-valore salvati nel browser dai siti web

- HTTP è un protocollo *stateless*
- Il sito **richiede** al browser di **salvare** uno o più cookie
- Il browser **allega** ad ogni successiva richiesta al server del sito tutti i cookie salvati dal sito

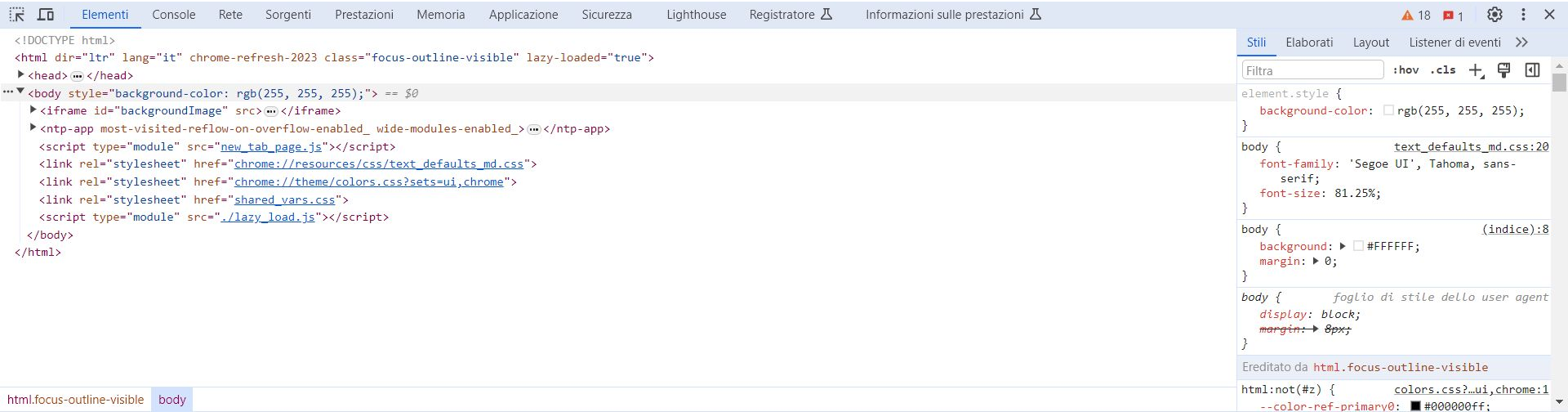

Client
1. Richiesta
GET http://www.example.com/ HTTP/1.1
2. Risposta + Set-Cookie
HTTP/1.1 200 OK
Set-Cookie: session-id=12345;
3. Richiesta + Cookie
GET http://www.example.com/ HTTP/1.1
Cookie: session-id=12345;
Server

Chrome DevTools

- Strumenti per sviluppatori di Chrome (o qualsiasi browser moderno)
- Debugging, Valutazione di performance, …, *geeking,* **hacking**
- Schede principali
  - Elementi
  - Console
  - Sorgenti
  - Rete
  - Applicazione

Chrome DevTools

- **DEMO TIME!**
- Da una pagina web aperta, premete il tasto **[F12]**



Libreria *requests* (Client HTTP per Python)

```python
import requests

# Richiesta GET
response = requests.get("https://jsonplaceholder.typicode.com/posts/1")
data = response.json()
print(data)

# Richiesta POST con dati in JSON
data = {"title": "foo", "body": "bar", "userId": 1}
response = requests.post("https://jsonplaceholder.typicode.com/posts", json=data)
print(response.json())
```

Libreria *requests* (Client HTTP per Python)

```python
import requests

# Aggiungere parametri
url = "https://jsonplaceholder.typicode.com/posts"
params = {"userId": 1}
response = requests.get(url, params=params)
print(response.json())

# Modificare gli header
headers = {"User-Agent": "my-app"}
response = requests.get(url, headers=headers)
print(response.status_code)
```

Libreria *requests* (Client HTTP per Python)

```python
import requests
from requests.auth import HTTPBasicAuth

# Autenticazione di base
auth = HTTPBasicAuth("user", "passwd")
response = requests.get("https://httpbin.org/basic-auth/user/passwd", auth=auth)
print(response.status_code)

# Download di un file
response = requests.get("https://www.example.com/sample.pdf", stream=True)
with open("sample.pdf", "wb") as file:
  for chunk in response.iter_content(chunk_size=1024):
    file.write(chunk)
```


THERE IS NOW A LEVEL ZERO.
getyarn.io


That's all Folks