TECH TALENT FACTORY
ISTITUTO TECNICO SUPERIORE

ISTITUTI TECNICI SUPERIORI
IT'S MY FUTURE

Cloud Administrator & Security Engineer

SCRIPTING & DEVOPS

SCRIPTING & BASH

andrea.scrivanti@gmail.com
Andrea Scrivanti

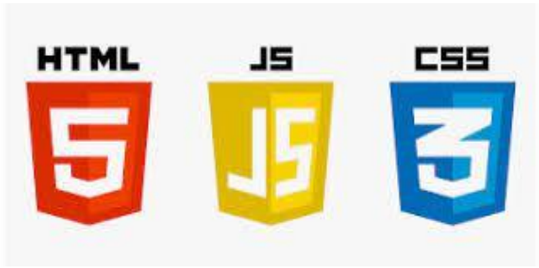
Gen-Mar 2023

Topics

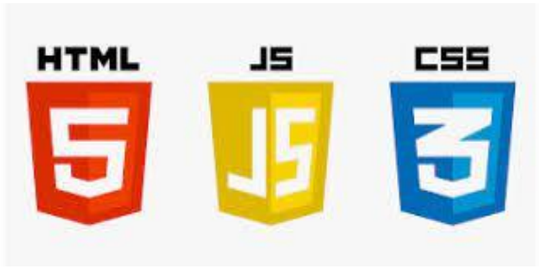
- Scripting & Bash
- Examples
- Test on AWS

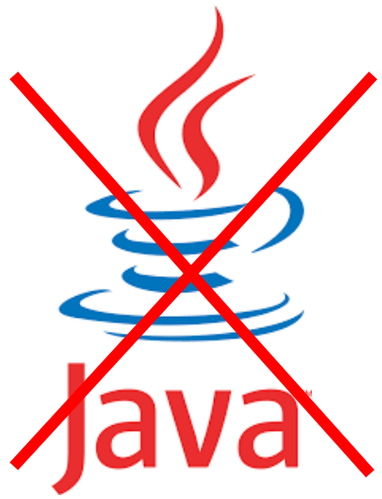
Linguaggi di scripting

Linguggi di scripting



Linguggi di scripting





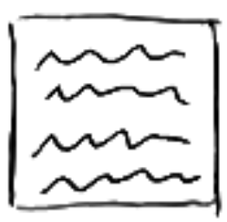
Proprietà in comune

- Linguaggio Interpretato
- Definizione di literal complessi
- Tipizzazione dinamica
- Programmazione interattiva tramite REPL

Linguaggio Interpretato

Source code:

hello.c



COMPILER

Machine code:

Program (also called binary, executable ...)

run the program

result

Source code:

hello.py

-> INTERPRETER -> result

Definizione di literal complessi

```
X = 10
y = "ciao"
```

```
X = [30,45,32]
```

```
X ={
'uno':1,
'due':2,
'tre':3,
....
}
```

```
andrea@idefix:~$ echo "Hello World!"
Hello World!
andrea@idefix:~$
```

Tipizzazione dinamica

```java
public class TestGson {

    public static void main(String[] args) throws Exception{
        String s = "Hello";
        s = 1;
    }

}
```

```
1    s = "Hello"
2    s = 1
3    print(s)
4
```

```
andrea@idefix:~$ X=10
andrea@idefix:~$ Y=2
andrea@idefix:~$ Z=$X+$Y
andrea@idefix:~$ echo "$Z"
10+2
andrea@idefix:~$ Z=$((X+Y))
andrea@idefix:~$ echo "$Z"
12
andrea@idefix:~$
```

Programmazione interattiva tramite REPL

```
andrea@idefix:~/workspaces_vscode/atlas-workspace$ python -i
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world!")
Hello world!
>>>
```

```
idefix:~$ X=10
idefix:~$ Y=2
idefix:~$ Z=$X+$Y
idefix:~$ echo "$Z"
10+2
andrea@idefix:~$ Z=$((X+Y))
andrea@idefix:~$ echo "$Z"
12
andrea@idefix:~$
```

Google scripting

All Images Videos Books News More

About 169,000,000 results (0.40 seconds)

A scripting language is **a programming language that employs a high-level construct t**
interpret and execute one command at a time. In general, scripting languages are easie
to learn and faster to code in than more structured and compiled languages such as C and

Elements Console Sources Network Performance Memory 4

top Filter Default levels 4 Issues: 4

```
> console.log("Hello world!");
  Hello world!                                    VM431:1
< undefined
>
```

Bash

Bash

- Bourne Again Shell (Estensione della shell dei sistemi unix)
- Linguaggio di programmazione (variables / conditions / loops)
- Case sensitive
- Set di comandi predefiniti

Bash – String manipulation

- VAR="questa.è.una.stringa.di.esempio"
- echo ${VAR#*.} # --> è.una.stringa.di.esempio
- echo ${VAR##*.} # --> esempio
-
- echo ${VAR%.*} # --> questa.è.una.stringa.di
- echo ${VAR%%.*} # --> questa
-
- echo ${VAR/st/ST} # --> queSTa.è.una.stringa.di.esempio
- echo ${VAR//st/ST} # --> queSTa.è.una.STringa.di.esempio

Bash – Condizioni

- if [-f prova.txt]
- then
- echo Il file esiste!
- else
- echo Il file non esiste!
- fi

Bash – Condizioni

- **-f nomefile**: ritorna vero l'oggetto esiste ed è un file.
- **-d directory**: ritorna vero se l'oggetto esiste ed è una directory.
- **-e file:** ritorna vero se l'oggetto esiste
- **str1 = str2**: ritorna vero se la str1 è uguale alla str2.
- **str1 != str2**: ritorna vero se la str1 è diversa dalla str2.
- **arg1 -eq arg2**: confronto numerico, ritorna vero se arg1 è uguale a arg2.
- **arg1 -ne arg2**: confronto numerico, ritorna vero se arg1 è diverso a arg2.
- **arg1 -lt arg2**: confronto numerico, ritorna vero se arg1 è minore a arg2 (**-le** per maggiore o uguale).
- **arg1 -gt arg2**: confronto numerico, ritorna vero se arg1 è maggiore a arg2 (**-ge** per maggiore o uguale).
- **espr1 -o espr2**: concatena più condizioni, uguale all'or, ritorna vero o se l'espr1 o se l'espr2 ritorna vero.
- **espr1 -a espr2**: concatena più condizioni, uguale all'and, ritorna vero se sia l'espr1 che l'espr2 ritornano vero.

Bash – Cicli

```
for i in 1 2 3 4 5
do
  echo "Numero $i"
done
```

```
#!/bin/bash
for (( i = 0 ; i <= 20 ; i += 2 )) ; do
  echo "i ha valore $i"
done
```

Bash – Cicli

```
until [[ -e procedi.txt ]] ; do
  sleep 3 # "dormi" per tre secondi
done
```

```
while [[ -e attesa.txt ]] ; do
  sleep 3 # "dormi" per tre secondi
done
```

Bash – functions

```
1 #!/bin/bash
2 sayHello() {
3     echo "Hello $1"
4 }
5
6 sayHello $1
7
```

Bash – Parameters

- $0: function name
- $1, $2 positional parameters
- $#: parameters count
- $@: all parameters
- $*: all parameters as one string

Bash – Basic commands

- df –h/free -h/ps aux/kill
- ls –lah
- cp/mv
- mkdir/rm -R
- cat
- head –n <n>
- tail –n <n>
- cd dir1 && echo $(pwd)
- cd dir1 || cd echo $(pwd)

More details? Use 'man' command!

Bash – grep

- Global search for the regular expression
- Various options: -i, -n, -v, -c
- grep “^hello” file1
- cat file1 | grep “^hello”
- grep “[a-e]” file1

Bash – sed

- Text stream editor
- sed 's/unix/linux/g' geekfile.txt
- cat geekfile.txt | sed 's/unix/linux/g'
- sed 's/unix/linux/p' geekfile.txt
- sed -n 's/unix/linux/p' geekfile.txt
- sed '/abc/d' filename.txt