---

2 Aprile 2025

ITPASS (COMMON Italy)

Python per Boomers

corso di programmazione in Python per soci/amici della associazione

---

* primo incontro: **Mercoledì** *2 Aprile* ore **21:00**

* successivamente: **Mercoledì ore 21:00**

ID riunione Zoom: **86417859959**

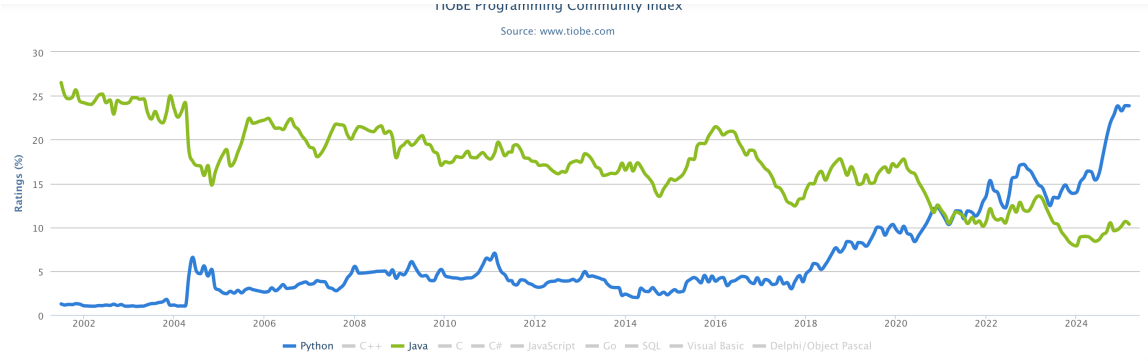
Codice d'accesso: **379791**

In [1]:
```
!python setallievi.py
```

In [ ]:
```
!python resoconto.py esempio
```

---

andamento



"niente è più pratico di una buona teoria"

Literate Programming

Intervista

Guido Van Rossum

Python

- high-level,
- general-purpose
- code readability (*indentation*)
- dynamically type-checked
- dynamically garbage-collected
- multiple programming paradigms:
- -- procedural,
- -- object-oriented and
- -- functional programming
- comprehensive standard library

"orienteering" in Python

- **map** (*mappa*) → `dir()`
- **compass** (*bussola*) → `type()`

```
5,  5.0,  True,  "itpass",  b"a\x7eb"
```

In [5]:
```
type(b"a\x7eb")
```

Out[5]:
```
bytes
```

```
[1, 2, "molti"],  ("A", "B", 3),  { 1: "ciao", 2: "tutti" },
{ "ciao", "a", "tutti" }
```

In [9]:
```
type({ "ciao", "a", "tutti" })
```

Out[9]:
```
set
```

---

int, float, bool, str, bytes

---

list, tuple, dict, set

---

variables and assignment

In [15]:
```
msg = "Ciao a tutti! Viva la città"
myBytes = msg.encode()
print(type(msg))
print(myBytes)
msg = 5.5
myBytes = msg
print(myBytes)
print(type(msg))
```

```
<class 'str'>
b'Ciao a tutti! Viva la citt\xc3\xa0'
5.5
<class 'float'>
```

tuple, str, bytes → *immutables*

In [19]:
```
msg = "Ciao a tutti!  Viva la città"
msg[11]
```

Out[19]:
```
'i'
```

In [18]:
```
msg = "Ciao a tutti!  Viva la città"
myBytes = msg.encode()
```

```
myBytes[-1]
```

Out[18]: 160

In [20]:
```
msg[11]= "*"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [20], in <cell line: 1>()
----> 1 msg[11]= "*"

TypeError: 'str' object does not support item assignment
```

In [21]:
```
msg = "Ciao a tutti!"
lista = list(msg)
lista[11]= "*"
msg = "".join(lista)
msg
```

Out[21]: 'Ciao a tutt*!'

In [ ]:
```
a = "Ciao"
metodi = dir(a)
for metodo in metodi:
    if not metodo.startswith('__'):
        print(metodo)
```

In [23]:
```
a = "ciao"
a.capitalize()
```

Out[23]: 'Ciao'

In [24]:
```
a = "ciao"
a.upper()
```

Out[24]: 'CIAO'

In [30]:
```
"un amico di nome %12.12s %s %04d" % ("Vincenzo", "Turturro", 15)
```

Out[30]: 'un amico di nome   Vincenzo Turturro 0015'

In [43]:
```
nome = ["Vincenzo", "Turturro"]
championships = 15
print(f"un amico di nome {nome[0]:<12s} {nome[1]} {championships:04d}")
print(f"un amico di nome {nome[0]:^12s} {nome[1]} {championships:04d}")
print(f"un amico di nome {nome[0]:>12s} {nome[1]} {championships:04d}")
```

```
un amico di nome Vincenzo     Turturro 0015
un amico di nome   Vincenzo   Turturro 0015
un amico di nome     Vincenzo Turturro 0015
```

In [4]:
```
class Numero:                # dove Java usava 'this', Python utilizza 'self'
    def __init__(self, valore = None) :
```

```
        if valore == None :
            pass
        else :
            self._valore = int(valore)

    def getValore(self):
        return self._valore

    def __repr__(self) :
        return "Ciao"
```

In [5]:
```
print(5, 6, 7, "Ciao", sep = "")
```

```
567Ciao
```

In [6]:
```
a = Numero(5.5)
```

In [7]:
```
print(a.getValore())
```

```
5
```

Sequence Containers Indexing

Boolean Logic

Statement Blocks

Conditional Statement (**if**)

Conditional Loop Statement (**while**)

Iterative Loop Statement (**for**)

Integer Sequences (**range**)

Generic Operations on Containers

Operations on Strings

Operations on Lists

Operations on Dictionaries

Operations on Sets

Function Definition and Call

Files and Exceptions

Module/Names Imports

Standard Library and non-Standard Libraries

Creating Classes and Methods

Inheritance
