Mistakepython

February 19, 2020

~~~ def useful_function(): for i in range(5): print("bella risposta: {}".format(i))

useful_function

if **name** == '**mane**': #non viene inserito in funzioni che potremmo voler importare!

[1]:
```
import time
```

~~~ while True: try: print("Messaggio infinito…") time.sleep(0.1) except: print("cazzarola continua a funzionare all'infinito")

Questa sintassi è sbagliata perchè un except generico intercetta solamente gli errori interni di python, non il CTRL+Z del terminale, lasciando il loop ancora all'infinito!

~~~ while True: try: print("Messaggio infinito…") time.sleep(0.1) except Exception: print("cazzarola continua a funzionare all'infinito")

In questo modo anche il SIGTERM del terminale fermerà il LOOP infinito!

~~~ import traceback try: something except: traceback.print_exc()

traceback ci permette di avere maggiori informazioni circa l'errore che ha bloccato il nostro codice.

```
with open("miofile") as f:
    names = f.readlines()[:1000000]
    names_set = set(names)
#prima funzione
def in_names():
    ret = []
    for i in range(100):
        ret.append(str(i) in names)
    return ret
#seconda funzione
def in_names_set():
    ret = []
    for i in range(100):
        ret.append(str(i) in names_set)
    return ret
```

La prima funzione, cercando in una lista, impiega 10 volte il tempo della seconda; ergo in questo caso meglio utilizzare dei dataset come SET o TUPLE… (vedere oggetti HASHABLE)

[6]:
```
import time
s = time.time()
for i in range(5):
    print("ciao amico {}".format(i))
    time.sleep(1)
print("il tempo di esecuzione del ciclo è stato {} secondi!".format(time.time()␣
 ↪- s))
```

```
ciao amico 0
ciao amico 1
ciao amico 2
ciao amico 3
ciao amico 4
il tempo di esecuzione del ciclo è stato 5.0053346157073975 secondi!
```

[9]:
```
def miaFunzione(nomi, classe = []):
    for name in nomi:
        classe.append(name)
    return classe

miaFunzione(["andrea", "mario"])
```

[9]:
```
['andrea', 'mario', 'andrea', 'mario']
```

[21]:
```
def miaFunzione(nomi, classe = None):
    classe = []
    for name in nomi:
        classe.append(name)
    return classe
```

[22]:
```
miaFunzione(["andrea", "mario"])
miaFunzione(["andrea", "mario"])
```

[22]:
```
['andrea', 'mario']
```

Questo succede perchè avendo utilizzato un oggetto mutabile ed essendo caricato in memoria, quando chiamiamo per la seconda volta la funzione questa aggiunge al vecchio contenuto il nuovo! Nel secondo caso, invece, abbiamo azzerato il contenuto all'inizio della chiamata e questo ci porta al risultato aspettato!

Nel caso in cui si voglia mettere un argomento con valore di default, questi dovrà essere un oggetto immutabile!

[ ]:
```

```