



Python ti odia quando sbagli?

No, vuole solo parlarti: guida allo StackTrace

Ovvero: come sopravvivere agli errori Python senza lanciare il PC dalla finestra ma usando Thonny (il nostro fedele alleato)

 III Liceo Scientifico Biella - Scienze Applicate

 Python Biella Group



Oggi imparerete ...

- Cos'è lo **stacktrace** e decifrarlo come veri hacker (spoiler: non è Matrix)
- A non urlare contro il computer quando il codice non funziona
- A capire che "ha funzionato al primo colpo" è un mito urbano



Cos'è uno StackTrace? 🤔

È il messaggio d'errore che Python ti lascia quando il tuo programma esplode.

Pensalo come:

- ✍️ Una lettera d'addio molto dettagliata
- 🗺️ Una mappa del tesoro (dove X = il tuo errore)
- 🚒 Il rapporto della polizia dopo l'incidente
- 💣 La scatola nera di un aereo che racconta tutto ciò che è successo prima del crash

Spoiler: Python è MOLTO specifico.



CHALLENGE #1: Il Concatenatore Confuso

```
def crea_messaggio_compleanno(nome, eta):  
    """Crea un messaggio personalizzato di compleanno"""  
    messaggio = "Buon compleanno " + nome + "!"  
    messaggio = messaggio + " Oggi compi " + eta + " anni!"  
    messaggio = messaggio + " Auguroni!"  
    return messaggio  
  
# Test  
nome_festeggiato = "Mario"  
anni = 18  
  
risultato = crea_messaggio_compleanno(nome_festeggiato, anni)  
print(risultato)
```




STACKTRACE #1

```
Traceback (most recent call last):
  File "compleanno.py", line 12, in <module>
    risultato = crea_messaggio_compleanno(nome_festeggiato, anni)
  File "compleanno.py", line 4, in crea_messaggio_compleanno
    messaggio = messaggio + " Oggi compi " + eta + " anni!"
TypeError: can only concatenate str (not "int") to str
```

Domanda: Perché Python si rifiuta di unire le stringhe?



Si legge dal BASSO verso l'ALTO! 

✗ La parte più bassa dello stacktrace (l'errore più recente) è dove si è verificato il crash, quindi è il primo punto da esaminare.

🧠 Le righe superiori ti aiutano a capire come l'errore si è propagato e ti forniscono un contesto utile per il debug. Puoi risalire nel "flusso" delle chiamate fino a trovare il punto in cui il programma ha preso una piega sbagliata.

Decodifichiamo il Messaggio

```
TypeError: can only concatenate str (not "int") to str
```

↑ INIZIA DA QUI: Il tipo di errore e cosa è successo

```
File "compleanno.py", line 4, in crea_messaggio_compleanno  
    messaggio = messaggio + " Oggi compi " + eta + " anni!"
```

↑ Dove è esploso tutto (file, riga, funzione)

```
Traceback (most recent call last):  
  File "compleanno.py", line 12, in <module>  
    risultato = crea_messaggio_compleanno(nome_festeggiato, anni)
```

↑ Come ci siamo arrivati (la "catena di eventi")



💡 SOLUZIONE #1

🐍 Problema: Stiamo cercando di concatenare una **stringa** con un **intero**!

Python non può fare "Oggi compi " + 18 perché non sa se vuoi:

- "Oggi compi 18" (conversione automatica)
- "Oggi compi " + "18" (tutto stringa)

Fix (3 modi):

```
# Metodo 1: Converti esplicitamente
messaggio = messaggio + " Oggi compi " + str(eta) + " anni!"

# Metodo 2: Usa f-string (il migliore!)
messaggio = f"{messaggio} Oggi compi {eta} anni!"

# Metodo 3: Usa format()
messaggio = messaggio + " Oggi compi {} anni!".format(eta)
```




MINI-CHALLENGE BONUS

Qual è l'output di questo codice?

```
numero = "10"  
risultato = numero * 3  
print(risultato)
```

- A) 30
- B) "101010"
- C) Errore
- D) "10 10 10"

Pensa prima di rispondere...



RISPOSTA BONUS

Risposta corretta: B) "101010"

In Python:

- `"abc" * 3` → `"abccabccabcc"` (ripete la stringa)
- `10 * 3` → `30` (moltiplica i numeri)

Morale: Il tipo di dato è TUTTO!

Se volevi 30, dovevi fare: `int(numero) * 3`



CHALLENGE #2: L'Indice Ribelle

```
def calcola_medie_mensili(temperature):  
    """Calcola la media delle temperature per ogni mese"""  
    medie = []  
  
    # Ci sono 12 mesi nell'anno  
    for mese in range(1, 13):  
        media_mese = temperature[mese] / 30 # Circa 30 giorni al mese  
        medie.append(media_mese)  
  
    return medie  
  
# Temperature totali per mese (ipotetico)  
temp_mensili = [450, 480, 520, 580, 650, 720,  
                780, 770, 690, 600, 510, 460]  
  
risultato = calcola_medie_mensili(temp_mensili)  
print("Medie mensili:", risultato)
```




STACKTRACE #2

```
Traceback (most recent call last):  
  File "temperature.py", line 16, in <module>  
    risultato = calcola_medie_mensili(temp_mensili)  
  File "temperature.py", line 7, in calcola_medie_mensili  
    media_mese = temperature[mese] / 30  
IndexError: list index out of range
```

Domanda: Perché Python non riesce ad accedere a `temperature[mese]` ?

Indizio: Da dove partono gli indici delle liste in Python?



Tips per Leggere gli StackTrace 💡

1. Non farti prendere dal panico (respira profondamente)
2. Leggi dal basso verso l'alto (sì, sempre)
3. Cerca il nome del TUO file (ignora roba di librerie esterne)
4. Guarda il numero di riga (poi vai a vedere QUELLA riga)
5. Leggi il messaggio finale (Python ti dice cosa è andato storto)



💡 SOLUZIONE #2

 Problema: `range(1, 13)` genera numeri da 1 a 12, ma gli indici della lista vanno da 0 a 11!

```
Lista:  [450, 480, 520, ..., 460]
Indici:  0    1    2    ...  11  ← Vanno da 0 a 11!
Mese:    1    2    3    ...  12  ← range(1,13) genera 1-12
                                     💀 temperature[12] NON ESISTE!
```

Fix:

```
# Metodo 1: Parti da 0
for mese in range(12): # 0, 1, 2, ..., 11
    media_mese = temperature[mese] / 30
# Metodo 2: Sottrai 1
for mese in range(1, 13):
    media_mese = temperature[mese - 1] / 30
```




FUN FACT: Off-by-one errors

Gli errori di "off-by-one" sono così comuni che hanno un nome proprio!

Le due regole fondamentali:

1. Gli indici in Python partono da 0
2. `range(n)` genera numeri da 0 a `n-1` (non fino a `n`!)

Trucco pro: Disegna la lista su carta con gli indici numerati. Old school, ma funziona sempre!



CHALLENGE #3: Il Divisore Zero

```
def calcola_media_classe(voti):  
    """Calcola la media dei voti di una classe"""  
    totale = 0  
  
    for voto in voti:  
        totale = totale + voto  
  
    media = totale / len(voti)  
    return media  
  
# Test con diverse situazioni  
print("Test 1 - Classe normale:")  
classe_a = [7, 8, 6, 9, 7, 8]  
print(f"Media: {calcola_media_classe(classe_a)}")  
  
print("\nTest 2 - Classe dopo l'influenza:")  
classe_b = [] # Tutti assenti!  
print(f"Media: {calcola_media_classe(classe_b)}")
```




STACKTRACE #3

Test 1 - Classe normale:

Media: 7.5

Test 2 - Classe dopo l'influenza:

Traceback (most recent call last):

File "media.py", line 18, in <module>

print(f"Media: {calcola_media_classe(classe_b)}")

File "media.py", line 8, in calcola_media_classe

media = totale / len(voti)

ZeroDivisionError: division by zero

Domanda: Cosa succede quando dividiamo per zero?

Risposta breve: Python va in panico 🌟



💡 SOLUZIONE #3

🐛 Problema: Lista vuota $\rightarrow \text{len(voti)} = 0 \rightarrow$ divisione per zero \rightarrow 🔥

Matematicamente: Non puoi dividere per zero! È impossibile!

Fix:

```
def calcola_media_classe(voti):  
    if len(voti) == 0: # Controllo PRIMA di dividere  
        print("Errore: nessun voto da calcolare!")  
        return 0 # 0 restituisci None, o solleva un errore  
    totale = 0  
    for voto in voti:  
        totale = totale + voto  
    media = totale / len(voti)  
    return media
```

Lezione: Sempre controllare le liste vuote prima di fare operazioni!



CASISTICHE COMUNI DI DIVISIONE PER ZERO

```
# ERRORE 1: Lista vuota
numeri = []
media = sum(numeri) / len(numeri) # 💀

# ERRORE 2: Contatore che rimane a zero
presenti = 0
media = totale_voti / presenti # 💀

# ERRORE 3: Input sbagliato
giorni = 0
consumo_giornaliero = consumo_totale / giorni # 💀
```

Regola: Prima di ogni divisione, chiediti: "Quel numero può essere zero?"



RECAP: Le Regole d'Oro

1. Leggi lo **stacktrace** dall'alto verso il basso (l'errore è in fondo)
2. Guarda il **TIPO** di errore:
 - `TypeError` → hai mescolato tipi incompatibili (`str + int`)
 - `IndexError` → indice fuori dai limiti della lista
 - `ZeroDivisionError` → hai diviso per zero
3. Gli **indici** partono da 0 (non dimenticarlo MAI!)
4. **Valida** gli input (liste vuote, divisioni per zero)
5. **Converti** i tipi esplicitamente (usa `str()`, `int()`, `float()`)



STRATEGIE DI DEBUG PRO

Quando il codice non funziona:

1. Leggi l'errore completo (non scappare!)
2. Guarda il numero di riga indicato
3. Aggiungi `print()` prima dell'errore:

```
print("Valore di x:", x)  
print("Tipo di x:", type(x))
```

4. Controlla i tipi delle variabili
5. Testa con input semplici prima di quelli complessi



CITAZIONI MOTIVAZIONALI

"Non ho fallito. Ho solo trovato 10.000 modi che non funzionano."

— *Thomas Edison (che non programmava in Python)*

"Debugging è come essere il detective in un giallo dove sei anche l'assassino."

— *Filipe Fortes*

"Il codice che funziona al primo tentativo è sospetto."

— *Ogni programmatore esperto*



DOMANDE?

Ricorda:

- Non esistono domande stupide
- Esiste codice che non funziona e stacktrace che non leggiamo

Trucco finale: Quando sei bloccato, prova a spiegare il problema a un amico (o a un papero di gomma). Spesso la soluzione arriva mentre spieghi!



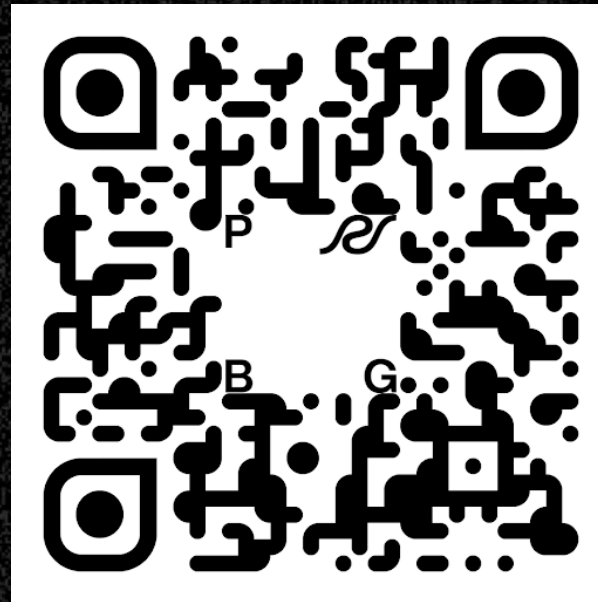
Conclusioni 🎓

Ricordate:

- Gli errori sono NORMALI (capita anche ai professionisti)
- Lo stacktrace è tuo AMICO (imparate a leggerlo!)
- Google è tuo ALLEATO (seriamente, usatelo)



Grazie per l'attenzione...



"C'è sempre qualcosa da imparare per migliorarci e crescere...insieme!"