🐛 Debug Wars: La Vendetta dello 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

PBG

📋 Oggi imparerete a...

- Decifrare gli **stacktrace** come veri hacker (spoiler: non è Matrix)
- Usare il **debugger** come un detective digitale 🕵️
- Non urlare contro il computer quando il codice non funziona
- Capire che "ha funzionato al primo colpo" è un mito urbano

PARTE I

Lo StackTrace: Il Diario Segreto di Python

"Caro diario, oggi l'umano ha sbagliato ancora..."

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. Troppo, forse.

Anatomia di uno StackTrace

```
Traceback (most recent call last):
  File "main.py", line 42, in <module>
    risultato = calcola_tutto()
  File "main.py", line 15, in calcola_tutto
    valore = dividi(10, 0)
  File "main.py", line 8, in dividi
    return a / b
ZeroDivisionError: division by zero
```

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

```
ZeroDivisionError: division by zero
```

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

```
  File "main.py", line 8, in dividi
    return a / b
```

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

```
  File "main.py", line 15, in calcola_tutto
  File "main.py", line 42, in <module>
```

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

I Classici Errori che Vedrete 

Errore	Traduzione Umana
`SyntaxError`	"Hai scritto male qualcosa, analfabeta!"
`NameError`	"Questa variabile non esiste, te la sei sognata?"
`TypeError`	"Non puoi sommare patate con carote"
`IndexError`	"Lista troppo corta, hai fatto il furbo?"
`IndentationError`	"Hai sbagliato ad indentare, controllare prego..."

🎯 SFIDA: Chi è il colpevole?

```
Traceback (most recent call last):
  File "studenti.py", line 23, in <module>
    media = calcola_media(voti)
  File "studenti.py", line 10, in calcola_media
    return sum(lista) / len(lista)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Domande:

1. Qual è il problema VERO?
2. In quale riga si trova?
3. Perché è successo?

Tempo: 30 secondi. Via!

Soluzione SFIDA #1 ✅

Il problema: Stai cercando di sommare numeri e stringhe!

Dove: Riga 10, funzione `calcola_media`

Perché: Probabilmente la lista `voti` contiene `["8", "7", 6]` invece di `[8, 7, 6]`

```
# MALE ❌
voti = ["8", "7", "6"]  # Stringhe!

# BENE ✅
voti = [8, 7, 6]  # Numeri!
```

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)

Golden Rule: Se l'errore non è chiaro, **copia-incolla su Google** (seriamente, lo fanno tutti i programmatori)

PARTE II

Il Debugger: Il Tuo Superpotere in Thonny

"Con grandi poteri derivano grandi... eh no, solo debugging"

Cos'è il Debugging? 

Debug = Togliere i "bug" (insetti) dal codice

> ***Fun Fact:*** *Nel 1947 un VERO insetto bloccò un computer. Grace Hopper lo rimosse e scrisse: "First actual case of bug being found"*

Oggi: I bug sono metaforici (ma altrettanto fastidiosi)

Il debugger: il tuo microscopio 🔬

Il debugger ti permette di:

- ⏸️ **Fermare** il programma durante l'esecuzione
- 👁️ **Guardare** cosa c'è dentro le variabili
- 👣 **Camminare** riga per riga nel codice
- 🕵️ **Capire** dove il programma impazzisce

In altre parole: È come mettere il programma in slow-motion

Thonny: Debug Mode 🎮

Come attivarlo:

1. Apri il tuo programma in Thonny
2. Clicca sull'icona 🐛 oppure premi `Ctrl+F5`
3. **BOOM!** Sei in debug mode

Oppure:

- Clicca sul numero di riga per mettere un **breakpoint** (punto rosso)
- Esegui normalmente (`F5`)
- Il programma si ferma al breakpoint

I Controlli del Debugger 🎮

Pulsante	Nome	Cosa fa
▶️	Run	"Vai fino al prossimo breakpoint"
⏭️	Step Over	"Esegui questa riga e vai alla prossima"
⏬	Step Into	"Entra dentro questa funzione"
⏫	Step Out	"Esci da questa funzione"
⏸️	Pause	"Fermati ORA!"

Breakpoint 🚩

Ovvero i checkpoint del codice: "Python, fermati QUI che devo controllare una cosa"

```
def calcola_sconto(prezzo, percentuale):
    # 🔴 BREAKPOINT QUI (riga 2)
    sconto = prezzo * percentuale / 100
    # 🔴 BREAKPOINT QUI (riga 4)
    prezzo_finale = prezzo - sconto
    return prezzo_finale
```

Quando il programma arriva al breakpoint:

- Si ferma
- Puoi vedere i valori delle variabili
- Puoi decidere cosa fare dopo

Le Variabili nel Debugger 👀

Quando sei in debug mode, Thonny ti mostra:

Pannello "Variables":

```
prezzo = 100
percentuale = 20
sconto = ??? (non ancora calcolato)
```

Puoi:

- Vedere i valori in TEMPO REALE
- Capire se sono corretti
- Scoprire dove il tuo cervello ha fallito

🎯 SFIDA: Debug Detective

```python
def trova_massimo(numeri):
    massimo = 0
    for num in numeri:
        if num > massimo:
            massimo = num
    return massimo

lista = [-5, -2, -10, -1]
risultato = trova_massimo(lista)
print(f"Massimo: {risultato}")
```

Problema: Dovrebbe stampare `-1`, ma stampa `0`

Missione: Usa il debugger mentalmente. Dov'è il bug?

Soluzione SFIDA 🎓

Il bug: `massimo = 0` all'inizio!

Se TUTTI i numeri sono negativi, nessuno è > 0

Quindi `massimo` resta 0 (SBAGLIATO!)

```python
# SOLUZIONE ?
def trova_massimo(numeri):
    massimo = numeri[0]  # Parti dal primo numero!
    for num in numeri[1:]:
        if num > massimo:
            massimo = num
    return massimo
```

Con il debugger: Avresti visto `massimo` restare 0 nel loop

PBG Metodo "barbaro" del "print"

A volte il debugger è troppo. Soluzione veloce che a volte è sufficiente:

```python
def calcola_media(voti):
    print(f"DEBUG: voti ricevuti = {voti}")  # ??

    totale = sum(voti)
    print(f"DEBUG: totale = {totale}")  # ??

    media = totale / len(voti)
    print(f"DEBUG: media = {media}")  # ??

    return media
```

Pro: Veloce e intuitivo

Contro: Dovrai cancellare tutti i print dopo (o dimenticartene e consegnare così al prof 😅)

Strategia di Debug Professionale 🎯

1. **Riproduci l'errore** (deve succedere SEMPRE, non "a volte")
2. **Leggi lo stacktrace** (ora sai come fare!)
3. **Formula un'ipotesi** (cosa PENSI sia il problema)
4. **Usa il debugger** (verifica la tua ipotesi)
5. **Testa la soluzione** (funziona davvero?)

Regola d'oro: Un bug alla volta. Non correggere 10 cose insieme.

SFIDA FINALE: Il Mistero del Codice

```python
def saluta_classe(studenti):
    for i in range(len(studenti)):
        nome = studenti[i]
        if nome == "Prof":
            print("Buongiorno, Professore!")
        else:
            print(f"Ciao {nome}!")

classe = ["Alice", "Bob", "Prof", "Diana"]
saluta_classe(classe)
```

Sembra giusto, ma... cosa succederebbe se `studenti = []` ?

Usa il debugger mentale!

Soluzione SFIDA FINALE 💡

Problema: Se `studenti = []`, il loop non trova problemi...

MA se modifichi il codice così:

```
def saluta_classe(studenti):
    for i in range(len(studenti) + 1):  # +1 qui!
        nome = studenti[i]  # ?? IndexError!
```

Con il debugger: Vedresti `i` diventare troppo grande

Soluzione Pythonica:

```
for nome in studenti:  # Più semplice!
    # ...
```

Errori Comuni dei Programmatori Junior 🤦

1. **Non leggere l'errore** ("Tanto non capisco" ❌)
2. **Cambiare codice a caso** ("Forse se metto +1 qui..." ❌)
3. **Non testare dopo ogni modifica** (poi hai 10 errori insieme ❌)
4. **Arrendersi troppo presto** ("È impossibile!" ❌)
5. **Non chiedere aiuto** (Orgoglio? Nel 2025? ❌)

Remember: Anche i professionisti debuggano per ore!

Tools Avanzati (Bonus) 🚀

In Thonny:

- **Highlight Syntax Errors:** Sottolinea errori di sintassi in rosso
- **Assistant:** Suggerimenti automatici (a volte utili!)
- **Variables view:** Vedi TUTTO quello che succede

Fuori da Thonny:

- `pdb` (Python Debugger integrato)
- Logging (invece di print)
- Unit tests (prevenire invece di curare)

La Filosofia del Debug 🧘

> *"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."*
> *— Brian Kernighan*

Traduzione: Scrivi codice SEMPLICE.
Il te-stesso-del-futuro ti ringrazierà.

Checklist del Buon Debugger ✅

- [ ] Ho letto TUTTO lo stacktrace?
- [ ] Ho capito QUALE riga genera l'errore?
- [ ] Ho usato il debugger per vedere le variabili?
- [ ] Ho testato la mia ipotesi?
- [ ] Ho cercato su Google/StackOverflow?
- [ ] Ho chiesto aiuto dopo 30+ minuti bloccato?
- [ ] Ho imparato qualcosa da questo errore?

Se hai 7/7: Sei pronto per il mondo reale! 🎓

Mini-Dizionario Debug 📖

- **Bug:** Errore nel codice
- **Debug:** Processo di rimozione degli errori
- **Debugger:** Tool per fare debug
- **Breakpoint:** Punto dove fermare l'esecuzione
- **Step Over:** Esegui riga corrente
- **Step Into:** Entra nella funzione
- **StackTrace:** Storia di come siamo arrivati all'errore
- **RTFM:** Read The Friendly Manual (leggete i messaggi d'errore!)

🎯 ESERCIZIO PER CASA

Trova i 3 bug in questo codice:

```
def calcola_media_classe(voti_studenti):
    totale = 0
    contatore = 0
    for voti in voti_studenti:
        for voto in voti:
            totale += voto
            contatore += 1
    return totale / contatore

classe_3A = [ [8, 7, 9],     [6, 6, 7],
              [],  # Studente assente!
              [9, 9, 10] ]

media = calcola_media_classe(classe_3A)
print(f"Media: {media}")
```

Hints per l'Esercizio 💡

1. **Cosa succede con lo studente assente?**
    - Il suo array è vuoto `[]`
    - Pensa al loop...
2. **Usa il debugger!**
    - Metti un breakpoint nel loop
    - Guarda `totale` e `contatore`
3. **Cosa succede se TUTTA la classe è assente?**
    - `contatore = 0`
    - `totale / contatore = ?` 💥

Conclusioni 🎓

Ricordate:

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

"Talk is cheap. Show me the code." — Linus Torvalds

Risorse Extra 📚

- **Documentazione Python:** docs.python.org
- **StackOverflow:** La Bibbia dei programmatori
- **Thonny Docs:** thonny.org
- **Real Python:** realpython.com/python-debugging-pdb/

E ricordate: Ogni errore è un'opportunità per imparare!

(O almeno così dicono per farci sentire meglio)

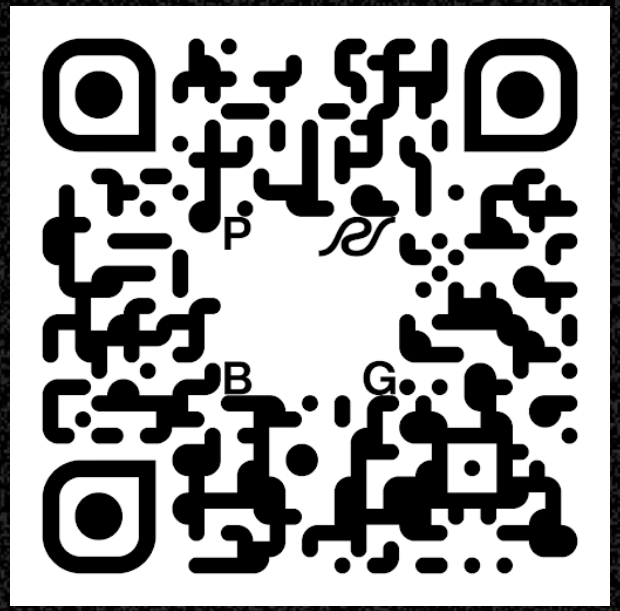
Fine! 🎉

Adesso andate a debuggare qualcosa!

P.S.: Se trovate un modo per debuggare la vita, fateci sapere 😅

```
while True:
    try:
        learn_from_mistakes()
    except Exception as e:
        print("Si cresce così!")
```

Grazie per l'attenzione...



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