



Oscar Cortez



# 5 PYTHON PITFALLS EVERY BEGINNER SHOULD AVOID



@OSKRGAB

# Indentation is More Than Aesthetic in Python

Unlike other languages that use brackets, Python uses whitespace indentation to define code blocks.

Forgetting this leads to the dreaded `IndentationError`.

```
● ● ●

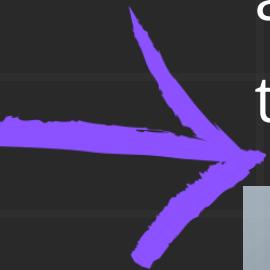
# Correct Indentation
def function():
    if True:
        print("Properly indented!")

# Incorrect Indentation leading to IndentationError
def function():
    if True:
        print("This will raise an error!")
```



# Lists vs. Tuples: Mutability Matters!

Lists are mutable and can be changed, while tuples are immutable and cannot. Confusing them can lead to unexpected errors.



```
# List: Mutable
my_list = [1, 2, 3]
my_list[0] = 9
print(my_list) # Output: [9, 2, 3]

# Tuple: Immutable
my_tuple = (1, 2, 3)
my_tuple[0] = 9 # This will raise a TypeError
```



# Variable Scope: Local vs Global

Variables defined inside a function are local to that function. Accessing them outside leads to `NameError`.

```
● ● ●

def my_function():
    local_var = 5
    print(local_var) # Works fine

my_function()
print(local_var) # NameError: name 'local_var' is not defined
```



# Dynamic Typing in Python: A Double-Edged Sword

Python's dynamic typing means no need to declare a variable's type. But beware, it can lead to type-related bugs if not careful.



```
● ● ●  
x = "100"  
y = 50  
print(x + y) # TypeError: can only concatenate str (not "int") to str
```



# Error Handling: The Safety Net of Your Code

Beginners often skip try-except blocks, leading to crashes when errors occur. Proper error handling is crucial.

```
● ● ●  
try:  
    file = open('non_existent_file.txt')  
    print(file.read())  
except FileNotFoundError:  
    print("File not found, please check the filename.")
```



@OSKRGAB



DONT FORGET  
TO LIKE, SHARE  
AND SAVE IF  
YOU LIKE THIS  
POST



OSKRGAB



@OSKRGAB