**Common errors in Python** – English

1. SyntaxError

If you write the syntax of your code incorrectly, you have written it wrong. In our case, it is Python syntax. Python has a syntax and you should write it exactly as it is.

Example: "pint" instead of "print", "swatch" instead of "switch", "wile" instead of "while".

1. IndentationError

Python has a code standard; you need to write your Python code using the correct spaces and gaps. If you put gaps and spaces in the wrong place, basically misplacing the code, an IndentationError will occur.

1. NameError

NameError occurs when you use a variable that has not been initialized. Example: C = C + 1. You need to declare your variable first and initialize it with C = 0.

1. ValueError

This happens when you pass the correct type of argument to a function, but use the wrong value.

Example: Python has a function that returns the square root of a number you give it.

So if you give the function -10, you are giving it the correct type of argument, which is a number, but you are giving it the wrong value, because a negative number does not have a square root.

1. UnboundLocalError

UnboundLocalError occurs when you try to use or print a variable that is not initialized, i.e., is not defined. This error is almost the same as NameError, but UnboundLocasError occurs more often when you use try and exception.

def example\_function():

try:

result = some\_operation()

except Exception as e:

print(&quot;An error occurred:&quot;, e)

print(result)

# Calling the function

example\_function()

1. TypeError

TypeError is when you incorrect use a variable, you try to do things with her that her type doesn’t allow, like printing a sentence with a number, when you have to print a string.

1. UnicodeError
2. "C:\Users\Eric\Desktop\beeline.txt"
3. "C:\\Users\\Eric\\Desktop\\beeline.txt"
4. ZeroDivisionError
5. FileNotFoundError
6. ModuleNotFoundError
7. MemoryError
8. PermissionError
9. IndexError
10. KeyError
11. AttributeErros