Exceptions, error handling - Advanced

Try - Except | Full control when having an error during execution

Once a .py file is executed, and somewhere along the way an error is thrown, the 'run' would immediately fail and stop.

What if we can have **full control** over our code,
and **prevent it from failing**. Exactly for that we can use the 'try-except' statement.

Code flow & Implementation

In this section we'll be discussing what happens when the code gets to a 'try-except' statement, and what are the logics behind it.

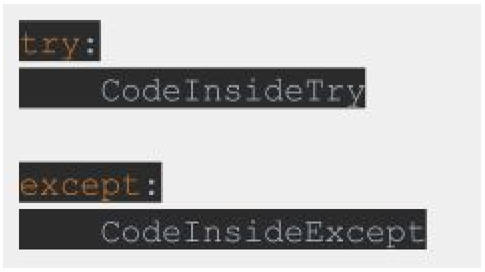
We'll also explain a code example, and break down the parts to mini-topics as always.

Code flow
Once we run a file that has 'try-except' statement -

1. Code execution will first run what's inside 'try'.

2. If no 'errors' are raised in the 'try' statement, the code will complete the execution within 'try', and will ignore the 'except' part completely. Whatever code appears after 'try', would also run.

3. But, if any error will be raised during the execution of the 'try' statement, the code will not fail or stop. The execution will instantly move to executing of what's inside 'except'.

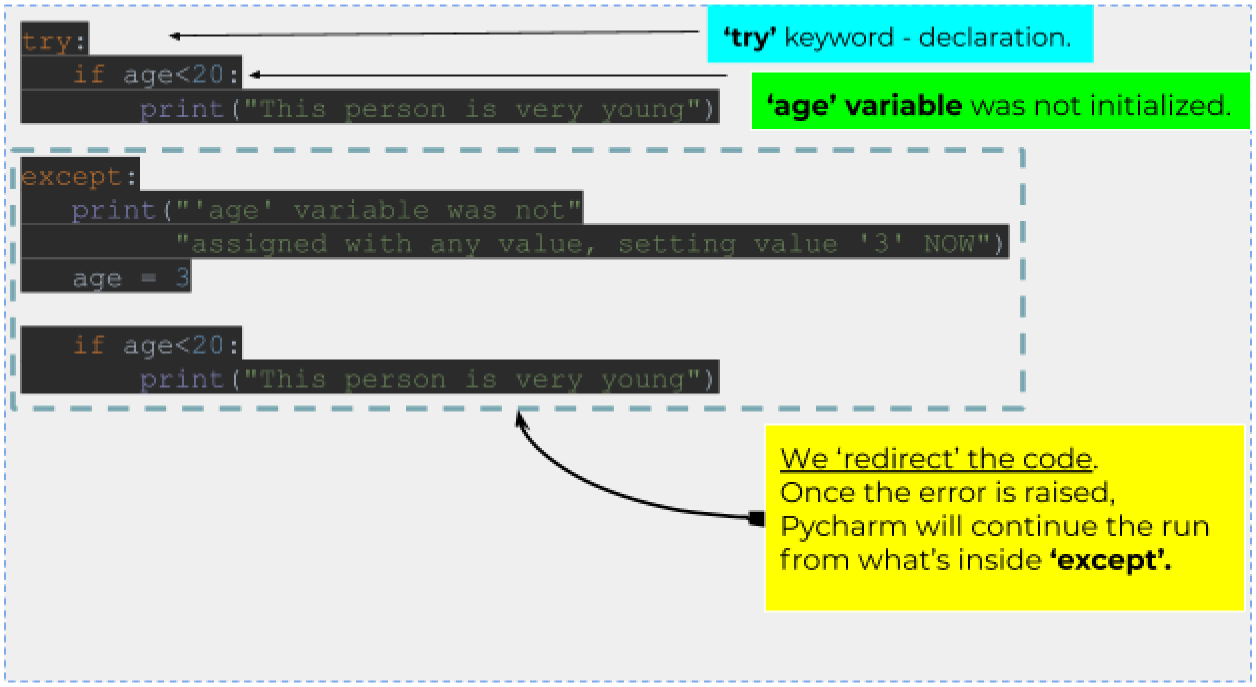


```
try:
    CodeInsideTry

except:
    CodeInsideExcept
```

Implementation example

The following code (*#1 in the list - from the previous lecture*) is a perfect example of how try-except gets us full control over flow of code. This chunk of code will cause a **'SyntaxError'** once executed.



```
try:
    if age<20:
        print("This person is very young")

except:
    print("'age' variable was not"
          "assigned with any value, setting value '3' NOW")
    age = 3

    if age<20:
        print("This person is very young")
```

'try' keyword - declaration.
'age' variable was not initialized.
We 'redirect' the code.
Once the error is raised,
Pycharm will continue the run
from what's inside **'except'.**