Advanced_flow_control

May 7, 2021

0.1 while ... else construct

```
while condition:
    executute_when_condition_is_true()
else:
    execute_when_condition_is_false()
```

The else branch is executed **if** or **when** the condition is false. Could happen the condition is false before entering the while clause and it jumps straight into the else clause. It could happen that after N cycles the while loop finishes, because the condition has changed to false, not because a break statement was found, and then the else clause is executed.

0.2 for ... else construct

```
for item in iterable:
    if match(item):
        result = item
        break
else: # nobreak
    # No match found
    result = None

# Always come here
print(result)
```

In the case of the for loop, the else section will be execute if the for loop finished iterating over the sequence (even if the sequence was empty). However, it will not be execute if the for loop finished because of a break statement.

0.3 Avoid using these obscure features

Most Python developers do not understand how the loop…else constructs work. So, better to avoid them, but how? For example, they can be extracted into a function such as:

```
def find_item(iterable, match):
    for item in iterable:
        if match(item):
            return item
    return None
```

0.4 try…else construct

```
try:
    # This code might raise an exception
    do_something()
except IOException:
    # IOException caught and handled
    handle_ioexception()
else:
    # No exception was raised
```

```
[ ]:
```