


Pattern: Nesting if-then in an else Pattern

- It is commonly useful to nest additional if-then-else statements inside of subsequent else-blocks
- Why? It allows us to choose one next step from many possible options.
 - "If this then do X, otherwise if that do Y, otherwise do Z."

```
if response == 0:  
    print("Very doubtful")  
else:  
    if response == 1:  
        print("Ask again later")  
    else:  
        print("It is certain")
```

Python has an "else if" construct for this purpose because it's so common and useful...

```
if response == 0:  
    print("Very doubtful")  
else:   
    if response == 1:  
        print("Ask again later")  
    else:  
        print("It is certain")
```

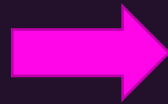


```
if response == 0:  
    print("Very doubtful")  
elif response == 1:  
    print("Ask again later")  
else:  
    print("It is certain")
```

1. Begin by deleting everything from the `se:` in `else:` to the start of the nearest `if`.

This is so common and useful, we tend to use simpler syntax for it...

```
if response == 0:  
    print("Very doubtful")  
elif response == 1:  
    print("Ask again later")  
else:  
    print("It is certain")
```

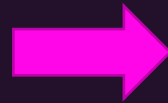


```
if response == 0:  
    print("Very doubtful")  
elif response == 1:  
    print("Ask again later")  
else:  
    print("It is certain")
```

2. Then, remove the extra indentation in the so that if/elif/else are all at the same level and their bodies are all one level in.

Using the **else-if** pattern is a change of *style* only. These two listings of code have the *exact same logic*.

```
if response == 0:  
    print("Very doubtful")  
else:  
    if response == 1:  
        print("Ask again later")  
    else:  
        print("It is certain")
```



```
if response == 0:  
    print("Very doubtful")  
elif response == 1:  
    print("Ask again later")  
else:  
    print("It is certain")
```

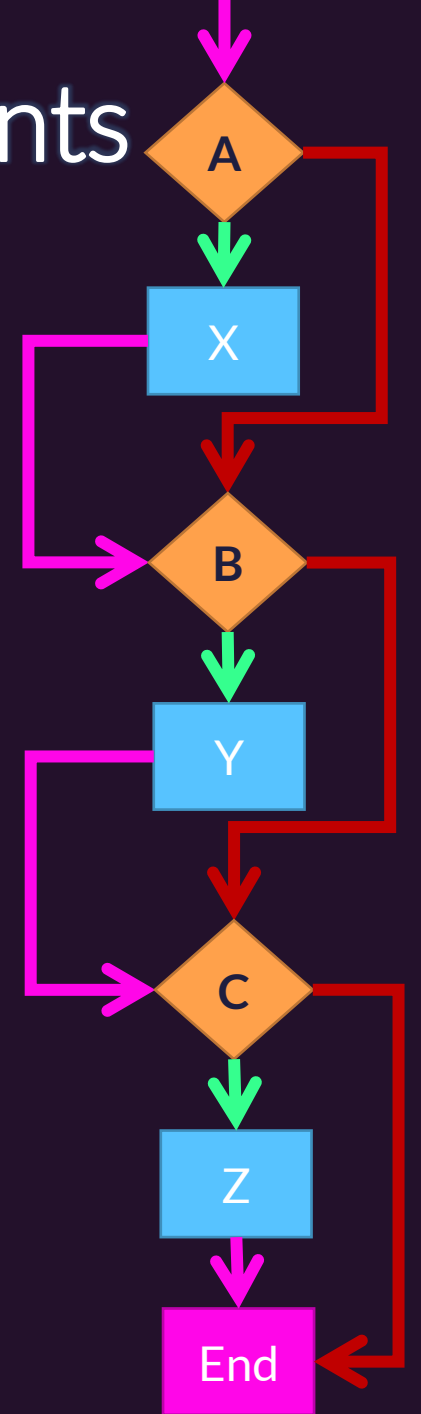
Notice the code is visually simpler and cleaner by using elif.

Note: You can have more than one elif branch in an if/elif/else statement. They are tested in order from top to bottom. As soon as one is true, its then block will evaluate and the rest of the elif/else branches skipped over.

Many, independent `if-then-else` statements

- When two or more `if-then-else` statements are *not* nested, they are independent statements of one another.
- Each boolean test expression will be evaluated.
- Notice in the diagram that there is a path through *every* block X, Y, Z.

```
if A:  
    print("X")  
  
if B:  
    print("Y")  
  
if C:  
    print("Z")  
  
print("End")
```



Tracing through `else-if` statements

- The previous slide does not apply to `else-if` statements *because...*
 - An `else-if` is a nested `if-then`
 - It is nested in the `else-block`
- Each boolean test expression will be evaluated until one evaluates to true. The rest are then skipped.
- Notice in the diagram that there is a path through *only one* outcome X, Y, Z.
- Useful when there are many possible next steps but you only want to choose one.

```
if A:  
    print("X")  
elif B:  
    print("Y")  
elif C:  
    print("Z")  
  
print("End")
```

