Making Decisions

Sequential Programming

Up until this point, *most* of the code we've written has gone sequentially (we took a little detour when working on the graphics stuff).

The Python Interpreter goes through our programs line by line, from top to bottom, one line at a time.

Sequential Programming

Up until this point, *most* of the code we've written has gone sequentially (we took a little detour when working on the graphics stuff).

The Python Interpreter goes through our programs line by line, from top to bottom, one line at a time.

```
num = int(input("How many? "))
correct_num = num >= 4 and num < 14
print("Your number was good: " + str(correct_num))
```

Sequential Programming

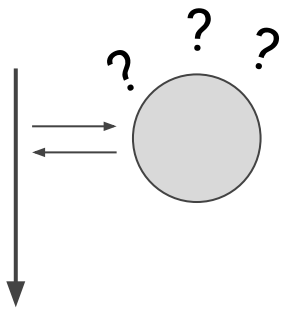
Up until this point, *most* of the code we've written has gone sequentially (we took a little detour when working on the graphics stuff).

The Python Interpreter goes through our programs line by line, from top to bottom, one line at a time.

```
num = int(input("How many? "))
correct_num = num >= 4 and num < 14
print("Your number was good: " + str(correct_num))
```

Adding a Detour

Today we're going to add the ability for our programs to take a detour under certain conditions - we're going to create **Non**-Sequential Programs!



if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Here we have the star of the show - the `if` itself. We need it if we want our programs to be able to make decisions!

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Here we have our `condition`. This must be a `boolean` typed value. What things have we learned about that can give us a `boolean`?

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Here we have our `condition`. This must be a `boolean` typed value. What things have we learned about that can give us a `boolean`?

- `boolean` variables
- Typecasting with `bool()`
- Logical Operators
- Comparison Operators

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Anytime we make an `if` statement, we need a colon after the `condition`.

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Lastly, we have the **body** of our `if`. When the `condition` is `True`, this code will run. When the `condition` is `False`, this code gets skipped over entirely.

if this or that

When we use an `if` statement, we can have the functionality of our program change based on different factors!

```
if condition:
    # ... do something …
    # ...
```

Lastly, we have the **body** of our `if`. When the `condition` is `True`, this code will run. When the `condition` is `False`, this code gets skipped over entirely. Notice the **indentation**! Everything you want to include in the `if` must be indented one level past the `if` itself.

if statement example

```
num = int(input("How many? "))
correct_num = num >= 4 and num < 14
if correct_num:
    print("Your number was good!")
```

if statement example

```
num = int(input("How many? "))
correct_num = num >= 4 and num < 14
if correct_num:
    print("Your number was good!")
```

if statement example

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
```

More Branching

What if I want something to happen when the `condition` for my `if` statement is `False`, though?

More Branching

What if I want something to happen when the `condition` for my `if` statement is `False`, though?

I can add an `else`!

`else` is basically a catch-all. We don't need to specify a `condition` for it, because it only runs when the `condition` for the `if` it's attached to is `False`.

How that look though?

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
else:
    print("Sorry, bad guess!")
```

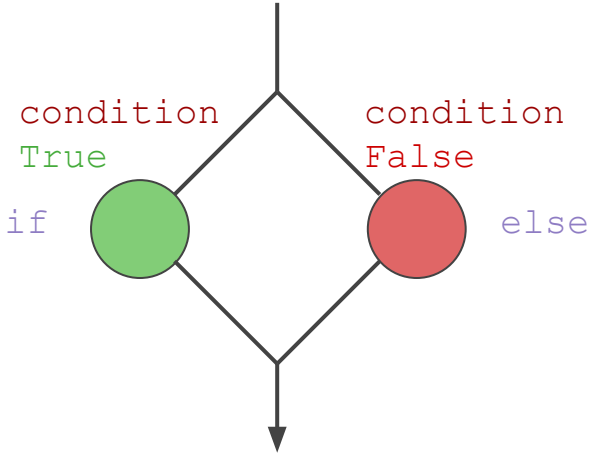
How that look though?

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
else:
    print("Sorry, bad guess!")
```

The `else` **must** be on the same level of indentation as the `if`, otherwise Python will not link them together properly!

Diverging Paths

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
else:
    print("Sorry, bad guess!")
```



Triverging Paths

If we want to add even more branches to our path, we can use an `elif` statement.

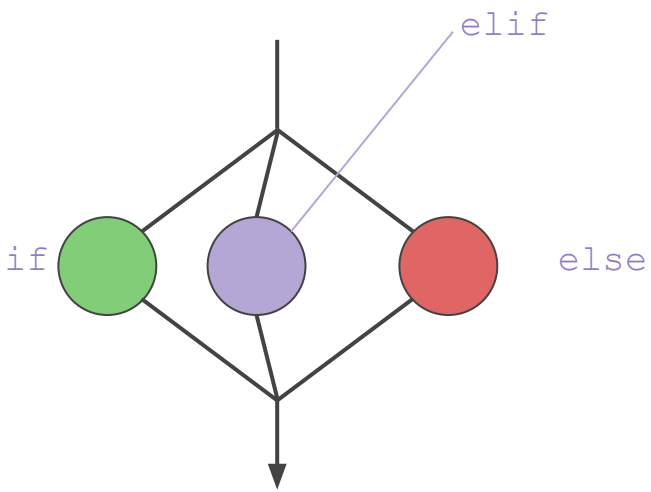
`elif`s belong in between `if` and `else`, and they combine their functionality!

An `elif` needs a `condition`, but it will only check its `condition` when the `condition` of the `if` above it was `False`.

We can have any number of `elif` statements tied to a single `if`, but **only one** is able to run!

Triverging Paths

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
elif num >= 14 and num < 16:
    print("Your number was alright.")
else:
    print("Sorry, bad guess!")
```



Quad...verging Paths?

```
num = int(input("How many? "))
if num >= 4 and num < 14:
    print("Your number was good!")
elif num >= 14 and num < 16:
    print("Your number was alright.")
elif num < 4 and num >= 0:
    print("Your number was fine.")
else:
    print("Sorry, bad guess!")
```