Conditionals

can code communities

What is Flow Control?

Flow Control allows us to manipulate how our code plays out.

Before, we would just write a series of steps that would be followed in a set order.

But with the topics being covered tonight we will be able to add variability to our code to make it more dynamic.

This will allow run to run differences in our code without us changing anything about it.

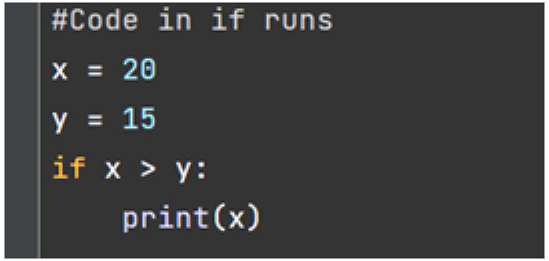
If Statements

If statements allow us to make dynamic code.

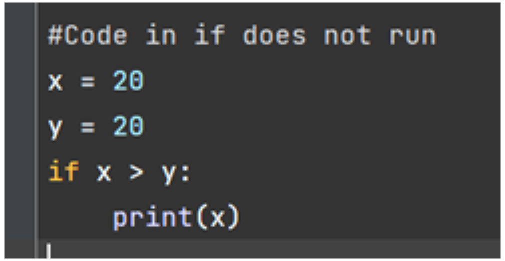
If statements take a boolean and chooses to perform the code inside based on whether the boolean is True or False.

Think: “if True, then code runs.”

We can also use boolean expressions (and, or, >, >= , <, <=, ==).



```python
#Code in if runs
x = 20
y = 15
if x > y:
    print(x)
```



```python
#Code in if does not run
x = 20
y = 20
if x > y:
    print(x)
```

Exercise

Write some code that prints “This is odd” if the user inputs an odd number.

(Hint: use an **if** statement)

Example:

User input: 7

```
This is odd
```

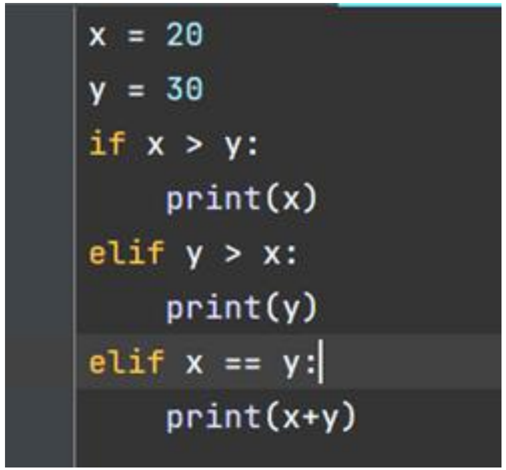
Else If (Elif) Statements

Allows us to add multiple conditionals to a single if structure.

Adds a second if statement so if the first one is False, Python then checks the condition of the second one.

If the if statement is True then the elif statement does not get evaluated

You can have multiple elif statements in an if structure.



```
x = 20
y = 30
if x > y:
    print(x)
elif y > x:
    print(y)
elif x == y:
    print(x+y)
```

Exercise

Add to your previous code so it prints “This is odd” if the user enters an odd number, and “This is even” if the user enters an even number.

(Hint: add an **elif** statement)

Examples:

User input: 7

```
This is odd
```

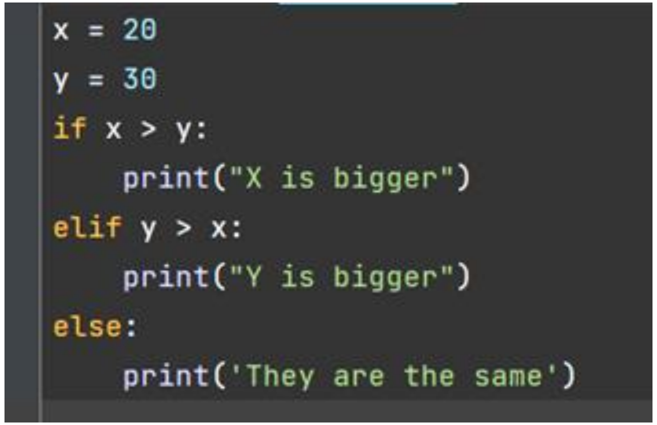
User input: 12

```
This is even
```

Else Statements

In simplest terms an Else statements is a catch-all at the end of a set of conditionals.

If all previous if and elif statements are False then the code in the Else runs.



```python
x = 20
y = 30
if x > y:
    print("X is bigger")
elif y > x:
    print("Y is bigger")
else:
    print('They are the same')
```

Exercise

Add to your previous code so if the user enters something that isn't an odd number or an even number, print "Unknown".

(Hint: add an **else** statement)

Examples:

User input: `8`

```
This is even
```

User input: `9`

```
This is odd
```

User input: `9.2`

```
Unknown
```

Remember to Indent

Whenever you have an if statement, an elif statement, or an else statement, everything within that statement must be indented (using one tab, or a consistent amount of spaces).

Once you stop indenting, you’re out of the if/elif/else block.

Indentation is essential in Python and relevant to many future topics.

Here is what PEP-8 has to say about indentation:
https://peps.python.org/pep-0008/#indentation

PEP-8 recommends using spaces over tabs, but it’s a debated topic among programmers, so use whatever you prefer - just be consistent.

Exercise

Write some code that takes in a string from the user and prints whether the string is a number, if it is a word, or something else.

Examples:

User input: `7`

```
This is a number
```

User input: `abcde`

```
This is a word
```

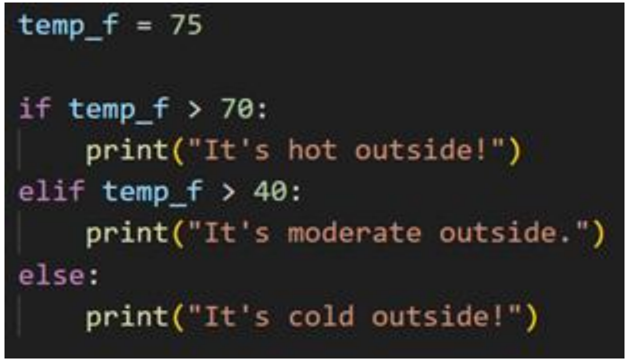
User input: `7!ab5`

```
This is something else
```

Chaining Conditionals

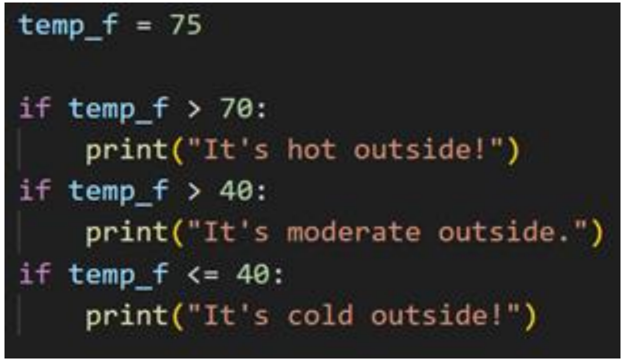
If you use `if/elif/else`, it forms a chain where all statements are mutually exclusive - only one of the blocks can be run.

If you put a bunch of if statements together like `if/if/if`, each one is evaluated separately, and multiple of them could be run.



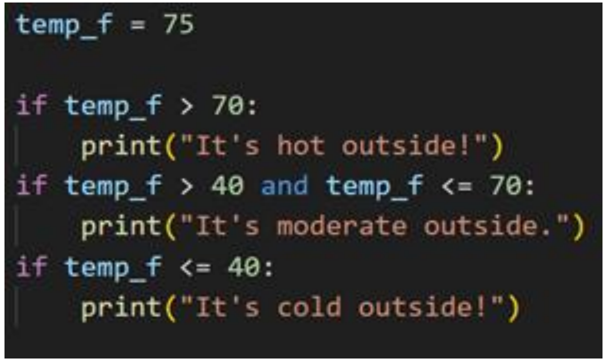
```python
temp_f = 75

if temp_f > 70:
    print("It's hot outside!")
elif temp_f > 40:
    print("It's moderate outside.")
else:
    print("It's cold outside!")
```



```python
temp_f = 75

if temp_f > 70:
    print("It's hot outside!")
if temp_f > 40:
    print("It's moderate outside.")
if temp_f <= 40:
    print("It's cold outside!")
```



```python
temp_f = 75

if temp_f > 70:
    print("It's hot outside!")
if temp_f > 40 and temp_f <= 70:
    print("It's moderate outside.")
if temp_f <= 40:
    print("It's cold outside!")
```

What is the difference between each of these code snippets?
Which ones are guaranteed to only print one of the options?
Which ones might print more than one?

Truth Values

The condition for an if or elif statement can be any boolean expression. It must be something that evaluates to True or False.

You never have to check if something `== True`. If it's True, then the if statement is automatically entered.

In fact, even non-booleans in Python have truth values. Anything that isn't empty, 0, None, or False, is considered True. So you can directly put it in your conditionals.

More info: https://docs.python.org/3/library/stdtypes.html#truth-value-testing

Boolean Operator Review

You can use a lot of different boolean operators in conditionals.

Here is the list:

`<`	Less Than
`<=`	Less Than or Equal To
`>`	Greater Than
`>=`	Greater Than or Equal To
`==`	Equals (value equality)
`!=`	Not Equals

`and`	Both True
`or`	One True
`not`	Opposite
`is`	Are they the same object (reference equality)
`in`	Inside of

Logical Operators

`and`, `or`, and `not` are a subset of boolean operators called logical operators.

- `and` takes two things and returns True if they are **both** True.
- `or` takes two things and returns True if **either one of them** is True, or both of them are true. (It is **inclusive or**.)
- `not` takes one thing and returns True if that thing is False, or vice versa - it returns the **opposite**.
  - `not a == b` is the same thing as saying `a != b`

Order of operations: not > and > or

- You can think of **and** as multiplication and **or** as addition

Nested Conditionals

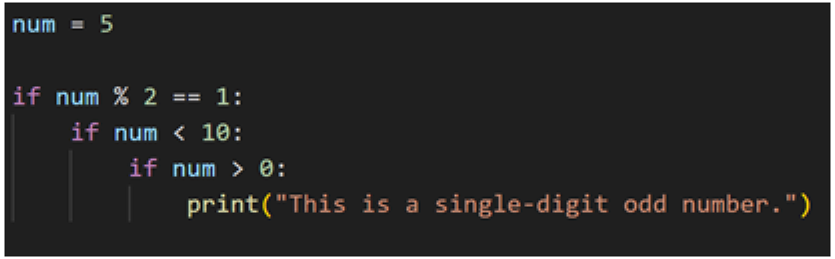
You can nest conditionals within other conditionals. Make sure the indentation always lines up. If you're in an if statement within another if statement, you should be 2 tabs indented.

Nesting conditionals allows for more complex code.

You can nest any type of conditional inside any other type of conditional (for example, an if statement within an else statement, an elif within an if statement, etc.)

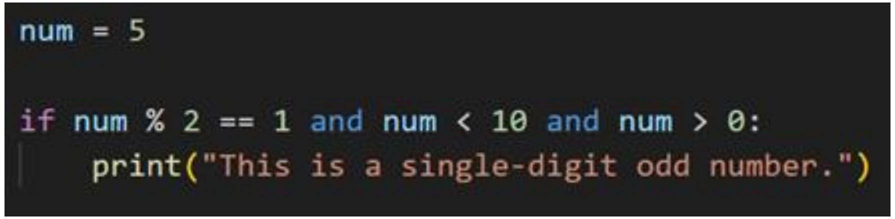
Nested Conditionals

Sometimes nested conditionals can be replaced by logical operators.



```python
num = 5

if num % 2 == 1:
    if num < 10:
        if num > 0:
            print("This is a single-digit odd number.")
```



```python
num = 5

if num % 2 == 1 and num < 10 and num > 0:
    print("This is a single-digit odd number.")
```

Discussion: What is the output of both of these code snippets?
Is it the same or different?

Exercise

You're working on a project to develop a login system for a website. The website requires the user to enter a username and password to log in. Write a Python program that checks whether the user entered the correct username and password.

- Create two variables called `username` and `password` and set them to any valid string values.
- Prompt the user to enter their username and password using the `input()` function.
- Use conditionals and logical operators to check whether the username and password entered by the user match the `username` and `password` variables.
- If they match, print "Login successful." If they don't, print "Incorrect username or password."

Resources

https://realpython.com/python-conditional-statements/

https://www.w3schools.com/python/python_conditions.asp