Strings

{string of individually
{accessible characters

- Python operates on white space alone, so no need for semicolons

{notes, terms, code}

- variables are generally always lowercase, separated by "_"
- you can use single or double quotes, it depends on whether you have single quotes in your desired message (for printing)
- triple quotes allow for a multi-line string.
- "len" prints out the length

of your desired messag.

Exampe Syntax:

```
message = "Hello World"
print(len(message))
```

Output - 11

- you can print at specific characters in a string by using the index feature

keep in mind that it starts from the 0th element.

Example syntax :

```
message = `Hello World'
print(message[0])
```

output : H

- you can easily access a range of characters as well, by using a colon.

Example Syntax:

```
message = `Hello World'
print(message[0:5])
```

the colon separates the starting and ending index

$\downarrow$

0 is the starting index, 5 is the ending index

The first index is inclusive and the second index is exclusive

meaning, the code above returns

Hello

when using the colon, you can opt to exclude a starting or finishing index

print(message[6:]) returns

world

String Methods

end &
• means at string

• lower(): returns any given string in all lowercase
• upper(): returns any given string in all uppercase
• count(): needs an argument,

returns how many times the argument is present in the string.

- find ( ): needs an argument, will return the index (or indices) of where the given argument is in your given string. If it's a word, it will return the index where a given word begins.

- replace : takes two arguments, what you want to replace (1), and the second argument, separated by a comma, is what we want to replace it with.

```
message.replace('World', '!')
```

for this, you can either create a

brand new variable, or, to save space, update the old one.

```
new_message = message.replace('world', '!')
```

OR

```
message = message.replace('world', '!')
```

simply adding strings together works, but that can get messy.

instead, use a formatted string!

```
greeting = 'Hello'
name = 'Meet'
```

message = '{}, {}. Welcome!'.format(greeting, name)

Output: Hello, Meet. Welcome!

"f strings" are used to make formatting as easy as possible.

message = f'{greeting}, {name}. welcome!'

Output: Hello, Meet. Welcome!

f strings allow you to write code within the placeholders.

Ex) f'{greeting.upper()}'

if you ever forget:

print(help(str))

will print out every possible method with a definition!