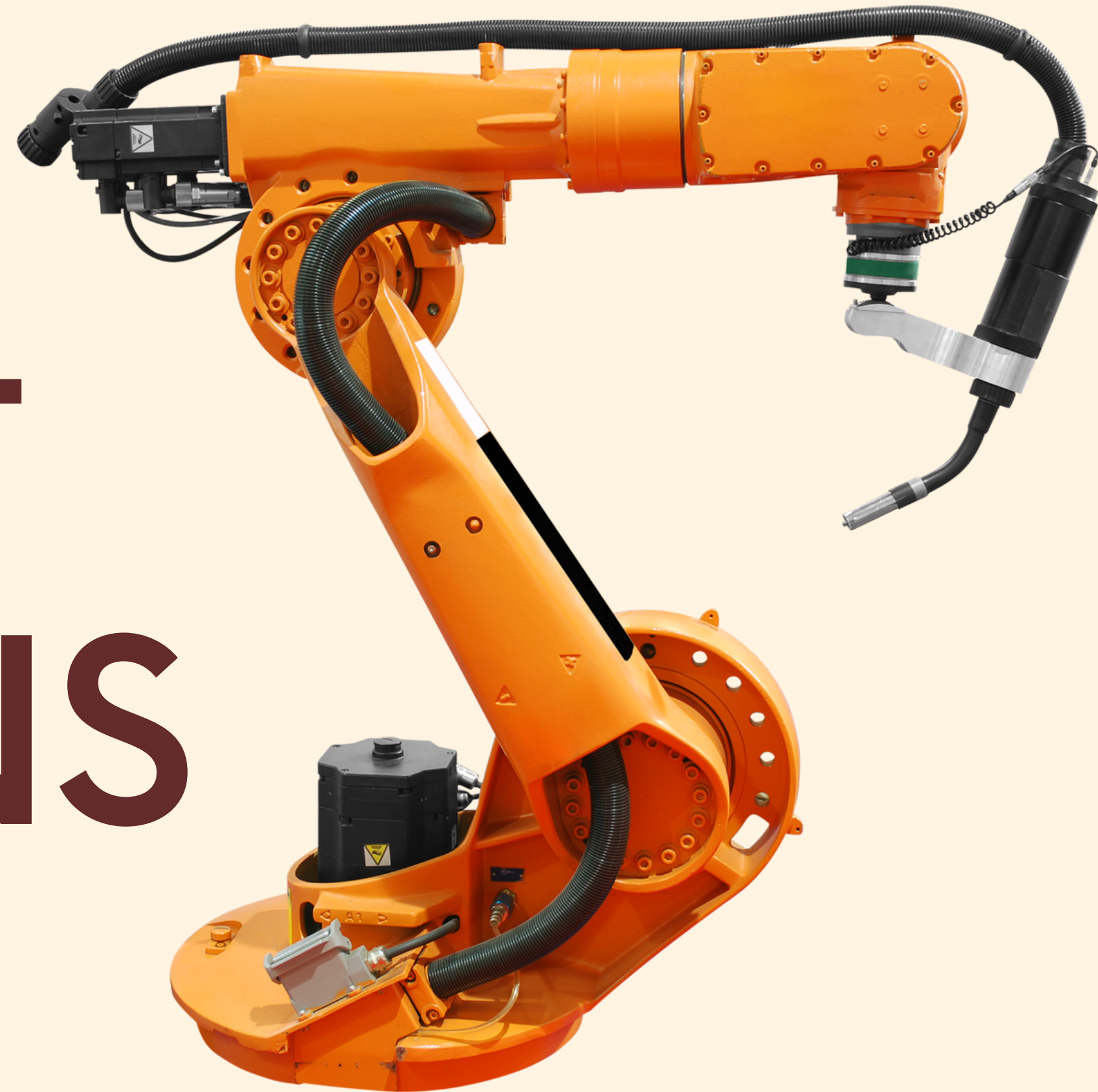


STRINGS + FUNCTIONS



STRINGS + FUNCTIONS



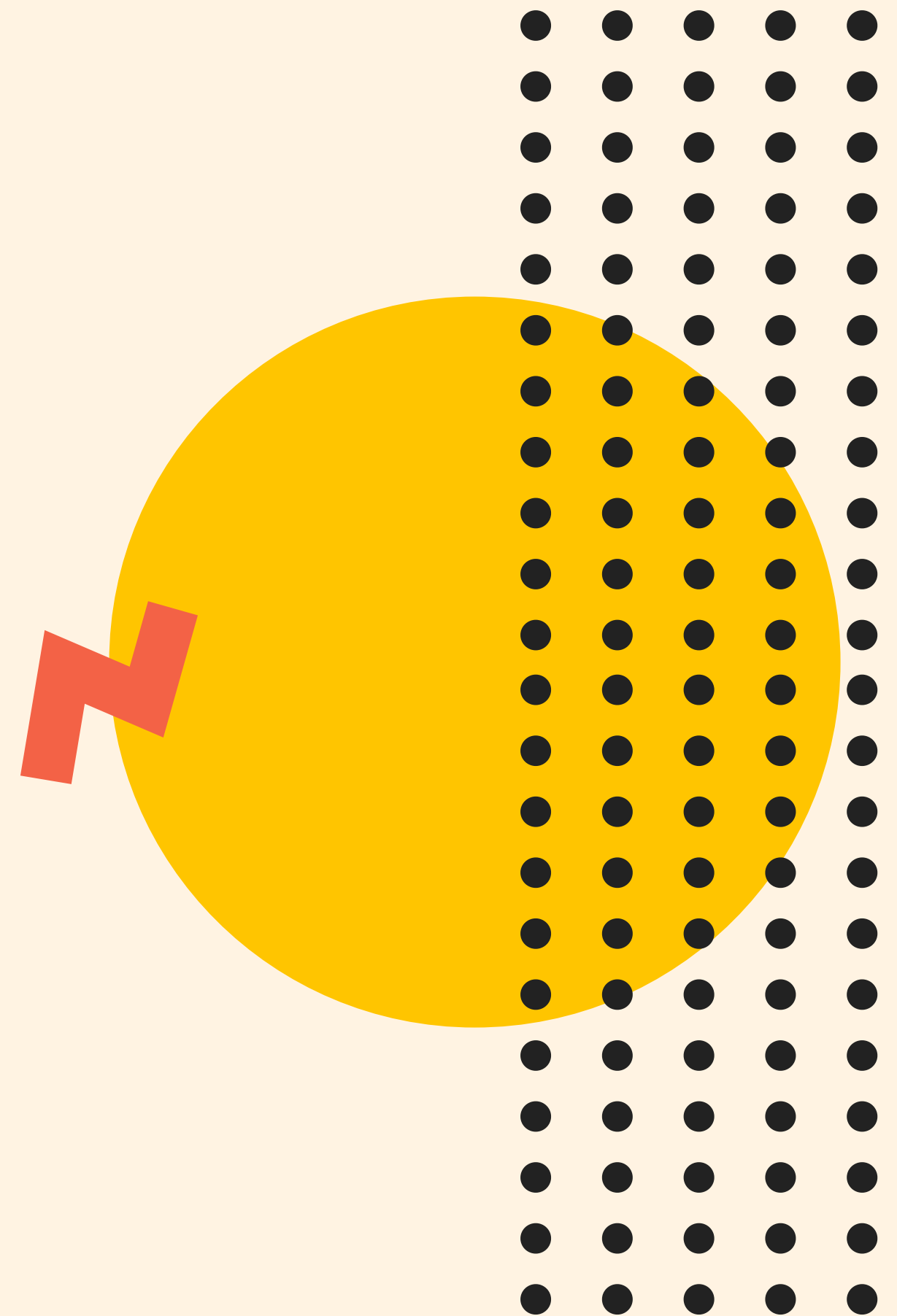
Functions

**FUNCTIONS ARE REUSABLE
ACTIONS THAT HAVE A NAME**



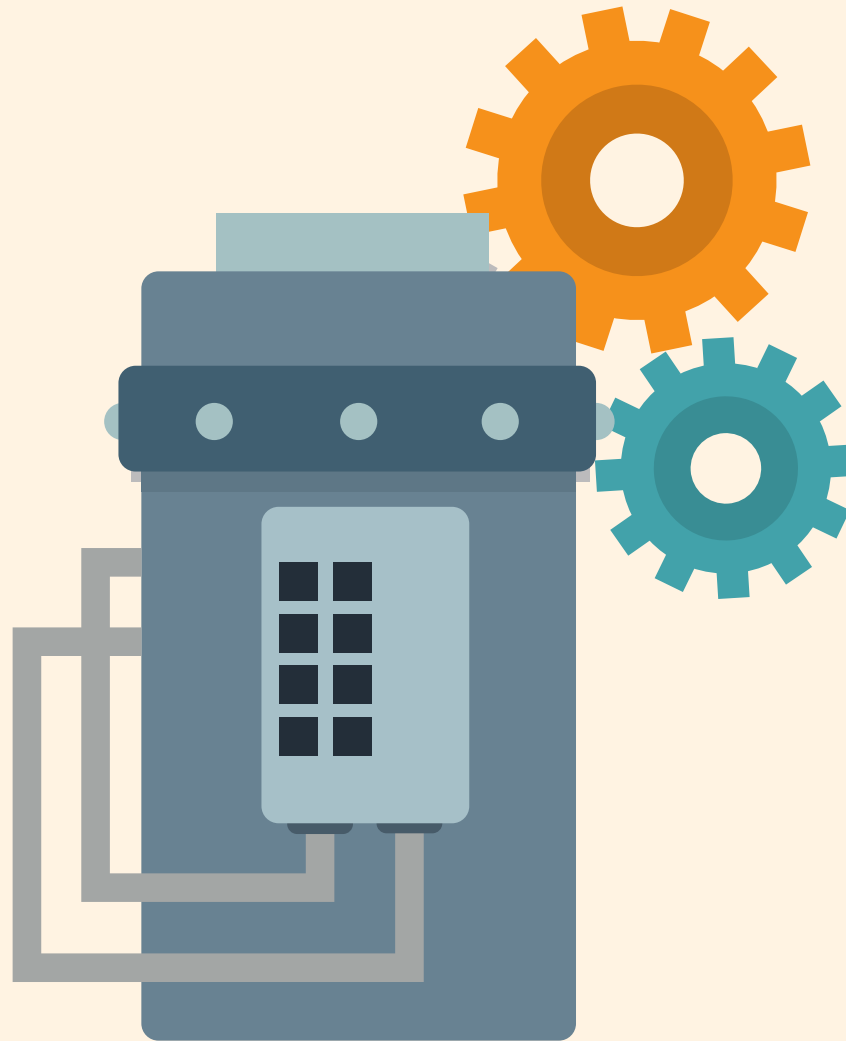
```
func_name()
```

TO EXECUTE A FUNCTION, WE USE PARENS ()



Inputs

`avg(20, 25)`



`avg(3, 2, 5, 6)`



Output



22.5



4

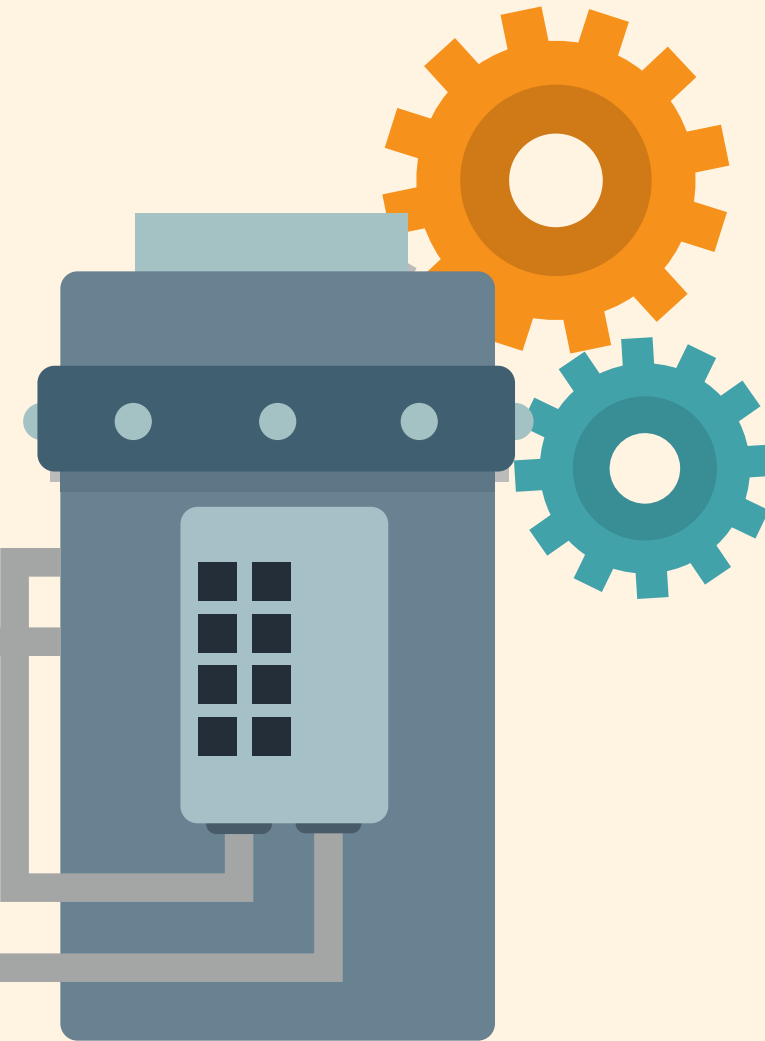
Inputs

Output

`get_weather(10003)`



`get_weather(92328)`



`"23 f"`

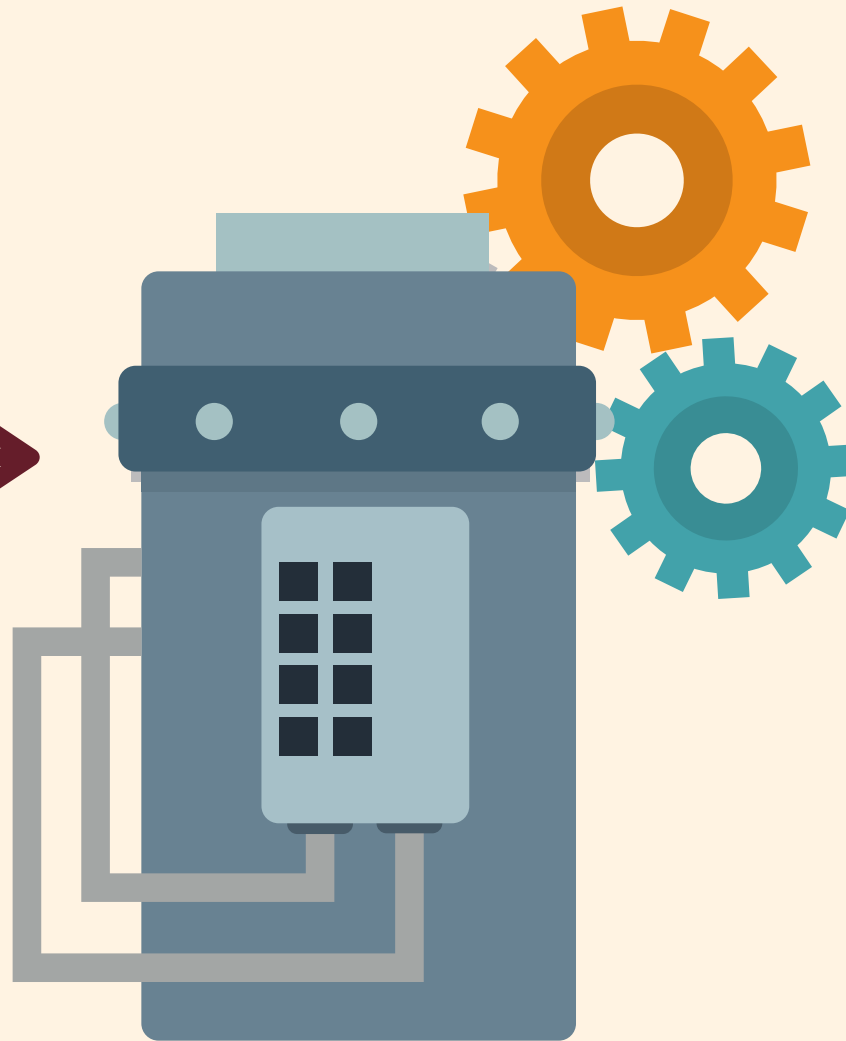
`"78 f"`

Inputs

`login('todd', 'jjkh2fj!d')`



`login('todd', 'kittycat')`



Output



False



True



Arguments

(Fancy word for inputs)

```
• • •  
>>> burger(bun,  
secret_sauce, lettuce,  
tomato, bacon, cheese,  
well_done)
```





Arguments

(Fancy word for inputs)



```
>>> burger(bun, tomato,  
bacon, cheese)
```









classmethod()

delattr()

dict()

dir()

filter()

getattr()

globals()

help()

id()

input()

isinstance()

issubclass()

iter()

len()

locals()

map()

max()

object()

open()

print()

property()

range()

reversed()

round()

setattr()

slice()

sorted()

sum()

type()

zip()

classmethod()

delattr()

dict()

dir()

filter()

getattr()

globals()

help()

id()

input()

instance()

issubclass()

iter()

len()

locals()

map()

max()

object()

open()

print()

property()

range()

reversed()

round()

setattr()

slice()

sorted()

sum()

type()

zip()


Length

The `len()` function will return the length of whatever item we pass to it. So far Strings are the only sequence we've seen, but soon we will see others!

```
● ● ●  
>>> word = "Chicken"  
>>> len(word)  
7
```

Input

The `input()` function will prompts a user to enter some input, converts it into a string, and then returns it. We can use it to gather user input in our programs



```
>>> age = input("how  
old are you?")
```

Type

The `type()` function accepts an input object and will return the type of that object

```
● ● ●  
>>> type("hi")  
<class 'str'>  
  
>>> type(55)  
<class 'int'>
```



Casting Types!

```
>>> int("12")  
12
```

```
>>> float("3.3")  
3.3
```

```
>>> str(44.5)  
'44.5'
```



Print

The `print()` function prints any arguments we pass to it to "standard output". It does not return anything.

A dark blue terminal window with rounded corners and three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal is a Python print statement: `>>> print("hello")`. The prompt characters are yellow, and the code is split into blue and green colors.

```
>>> print("hello")
```



f strings

f-strings are an easy way to generate strings that contain interpolated expressions. Any code between curly braces `{ }` will be evaluated and then the result will be turned into a string and inserted into the overall string.



```
>>> f"there are {24*60*60} seconds in a day"  
"there are 86400 seconds in a day"
```



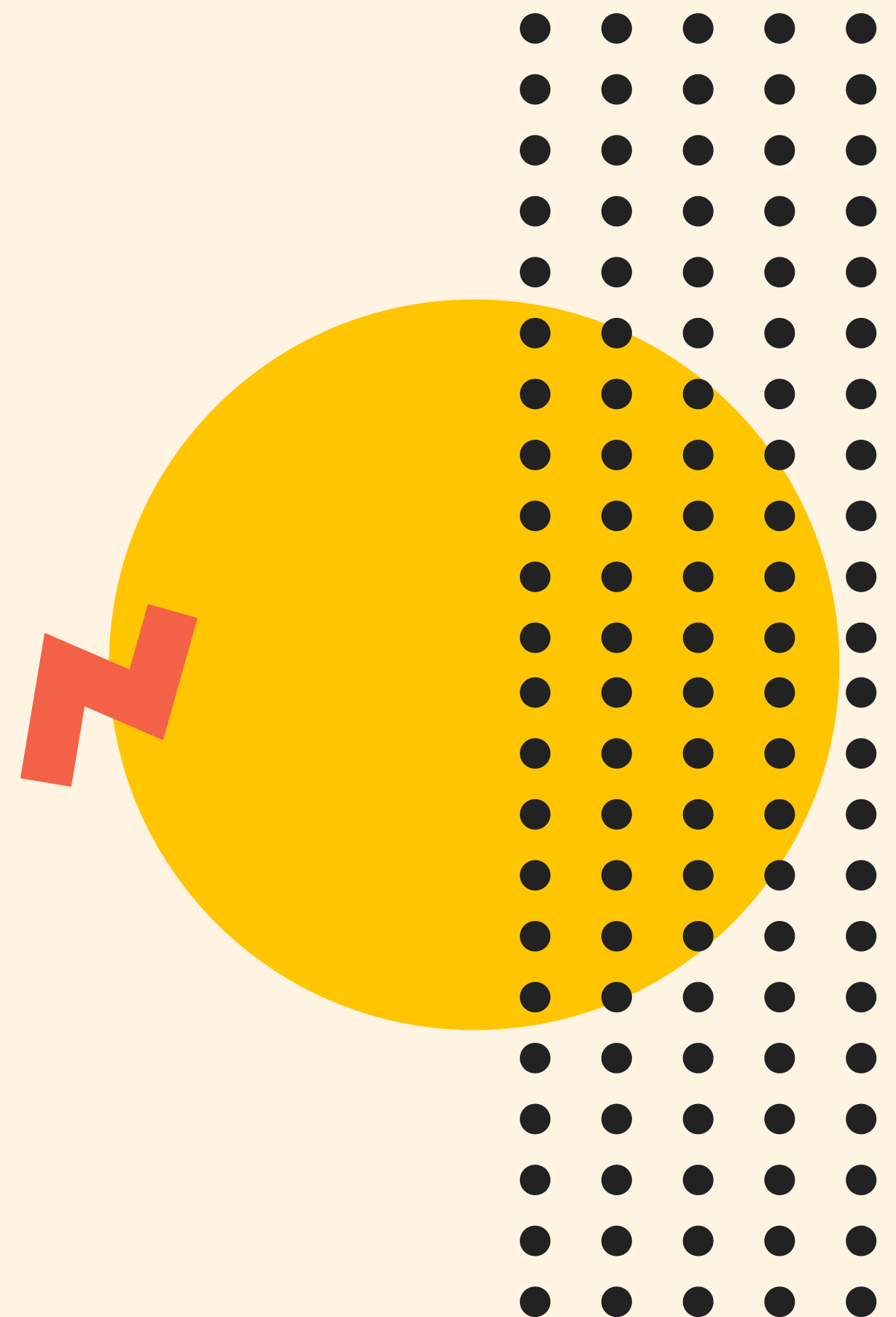
Methods

**METHODS ARE FUNCTIONS
THAT "LIVE" ON OBJECTS**



```
thing.method()
```

METHODS AUTOMATICALLY HAVE ACCESS TO THE
OBJECT THEY ARE CALLED ON.



thing.method()

String Methods

`str.capitalize()`

`str.endswith()`

`str.find()`

`str.join()`

`str.lower()`

`str.lstrip()`

`str.removeprefix()`

`str.removesuffix()`

`str.replace()`

`str.rfind()`

`str.rindex()`

`str.rjust()`

`str.split()`

`str.startswith()`

`str.strip()`

`str.swapcase()`

`str.title()`

`str.upper()`



Capitalization Methods



```
>>> msg = "Hello world"
```

```
>>> msg.capitalize()
```

```
Hello world
```

```
>>> msg.upper()
```

```
HELLO WORLD
```

```
>>> msg.lower()
```

```
hello world
```

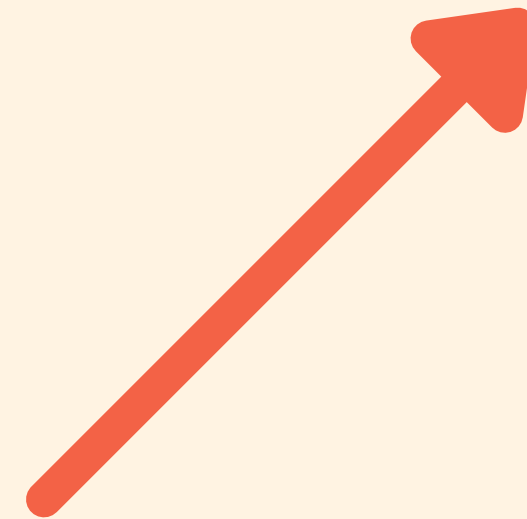


```
str.upper()
```



**accepts no
arguments!**

```
str.strip([chars])
```



chars is optional

`strip()`

Strips space characters

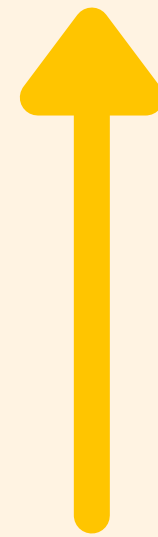
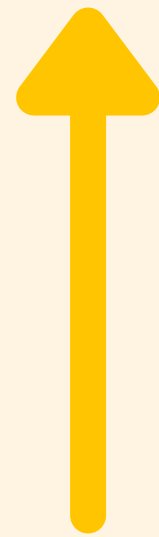
`strip('-')`

Strips '-' characters

`strip('= -')`

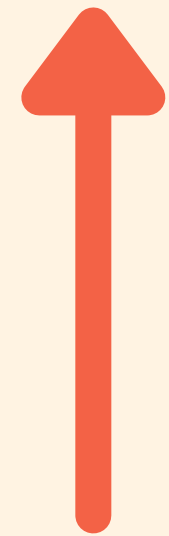
Strips '=' and '-' characters

```
str.replace(old, new, [count])
```



old and new are required

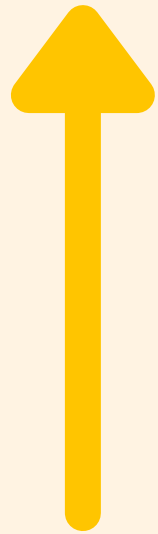
```
str.replace(old, new, [count])
```



count is optional

```
find(sub[, start[, end]]) -> int
```

`find(sub[, start[, end]]) -> int`



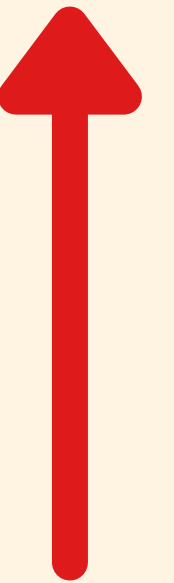
sub is a required argument

`find(sub[, start[, end]]) -> int`

anything in `[]` is optional



`find(sub[, start[, end]]) -> int`



find returns an integer

`find("c")` returns index where "c" is first found

`find("c", 5)` returns index where "c" is first found, after index 5

`find("c", 5)` returns index where "c" is first found, after index 5

method chaining



Strip Methods



```
>>> msg = "...end..."
>>> msg.strip('.')
end
>>> msg.lstrip('.')
end...
>>> msg.rstrip('.')
...end
```





Find & Index



```
>>> msg = "Cat in a hat"
```

```
>>> msg.find('a')
```

```
1
```

```
>>> msg.rfind('a')
```

```
10
```

```
>>> msg.index('a')
```

```
1
```





Replace & Count



```
>>> msg = "Hot dog"
>>> msg.replace('o', 'u')
Hut dug
>>> msg.replace('o', 'u', 1)
Hut dog
>>> msg.count('o')
2
```

