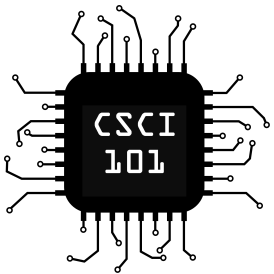
Functions



Some Functions you Already Know

We've already been using functions for a while now. Here are some you already know:

- `print(value, ...)` – Writes all of its arguments to the console on a single line separated by spaces.

Some Functions you Already Know

We've already been using functions for a while now. Here are some you already know:

- `print(value, ...)` – Writes all of its arguments to the console on a single line separated by spaces.
- `input(prompt)` – Prompts a user for input and returns the string they typed. `prompt` is an optional parameter.

Some Functions you Already Know

We've already been using functions for a while now. Here are some you already know:

- `print(value, ...)` – Writes all of its arguments to the console on a single line separated by spaces.
- `input(prompt)` – Prompts a user for input and returns the string they typed. `prompt` is an optional parameter.
- `len(sequence)` – Returns the length of the provided sequence (can be a string, list, set, or more).

Making Our Own Functions

Python allows us to make our own functions. Here is a simple example:

```
def greet(name):
    print("Nice to meet you,", name)
```

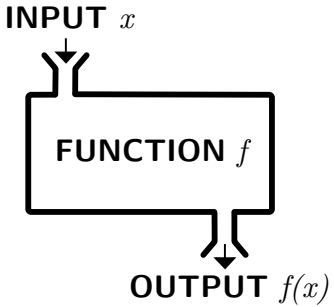
We can then call the function like any other function:

```
greet("Alice")
greet("Bill")
```

```
Nice to meet you, Alice
Nice to meet you, Bill
```

Why Make Functions?

- With functions, we can clean up repetitive code by combining common features
- With functions, we can hide the implementation to the programmer to simplify programming



Key Point

Functions provide the power of **abstraction**. This allows us to preform similar operations without needing to use separately coded parts.

Indentation Still Determines Scope

```
def my_fun():
    print("Inside my_fun!")
    print("When does this print?")
my_fun()
```

```
def my_fun():
    print("Inside my_fun!")
print("When does this print?")
my_fun()
```

Type in both Python programs and compare the output.

Variables in a Function

Variables inside a function are accessible only to that function call. Attempts to access those variables outside the function will result in an exception.

```
def hello(name):
    sentence = "Hello, " + name + "!"
    print(sentence)

hello("John")
print(sentence)      # this will cause an error
```

Returning from a Function

Just like `input` returns a string, your functions may return a value as well. Consider the following simple example:

```
def add_em(x, y):
    result = x + y
    return result
```

Calling `print(add_em(12,15))` would print:

```
27
```

Returning Immediately Ends a Function Call

As soon as a `return` statement is encountered, the function call will immediately return and will not continue to execute.

What will this example print? Type the code into Python to check your answer.

```
def add_and_print(x, y):
    result = x + y
    return result
    print("The sum is", result)

add_and_print(12, 15)
```

Practice: Doubling Function

Practice by writing a function that takes a list as its parameter, doubles all elements in the list, and returns the doubled list.

```
def double_elements(my_list):
    # your code here

print(double_elements([1, 2, 3, 4]))
print(double_elements([7, 14, 21, 28, 35]))
print(double_elements([42, 42, 42]))
print(double_elements([]))
```