香港浸會大學
HONG KONG BAPTIST UNIVERSITY

Functions

JOUR7280/COMM7780

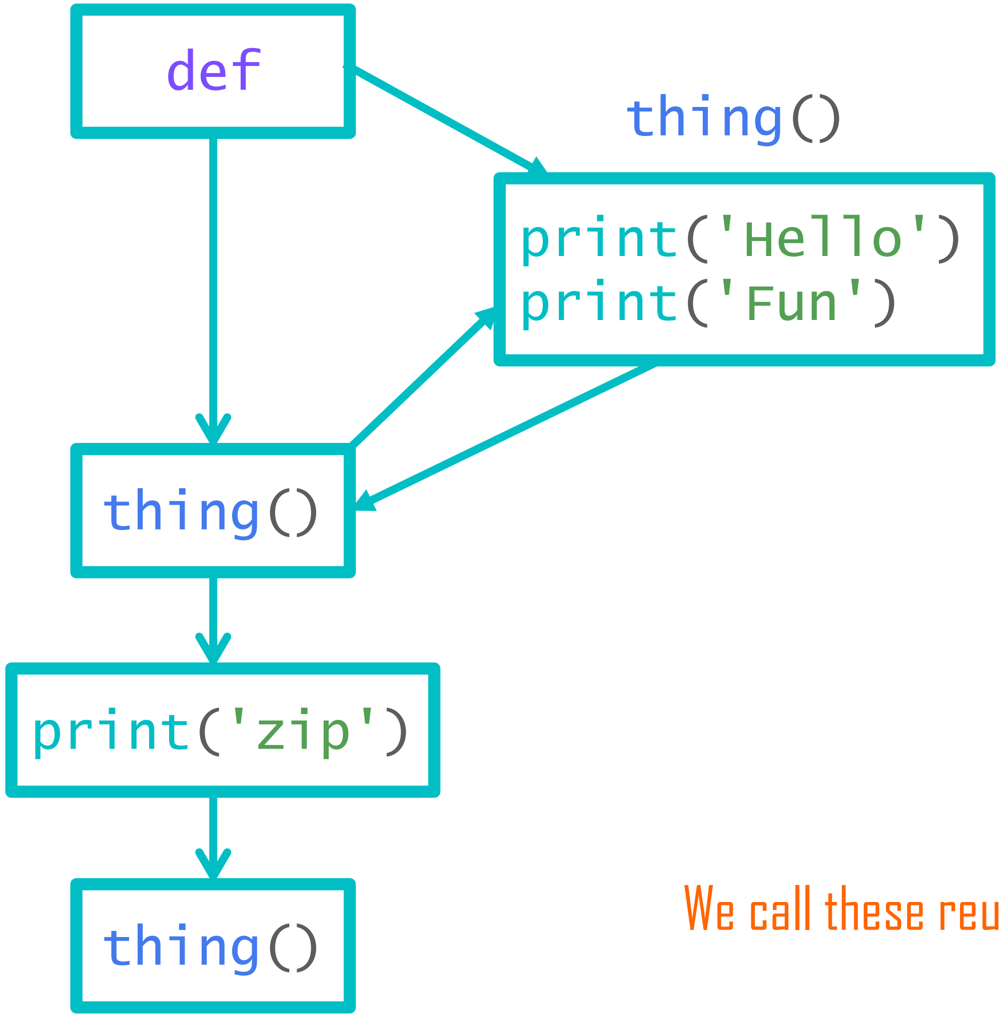
Big Data Analytics for Media and Communication

Instructor: Dr. Xiaoyi Fu

Four Patterns for Code

- Sequential
- Conditional
- Iterations
- Store and reuse
    - DRY: don't repeat yourself

Stored (and reused) Steps



Program

```
def thing():
    print('Hello')
    print('Fun')

thing()
print('zip')
thing()
```

Output

```
Hello
Fun
zip
Hello
Fun
```

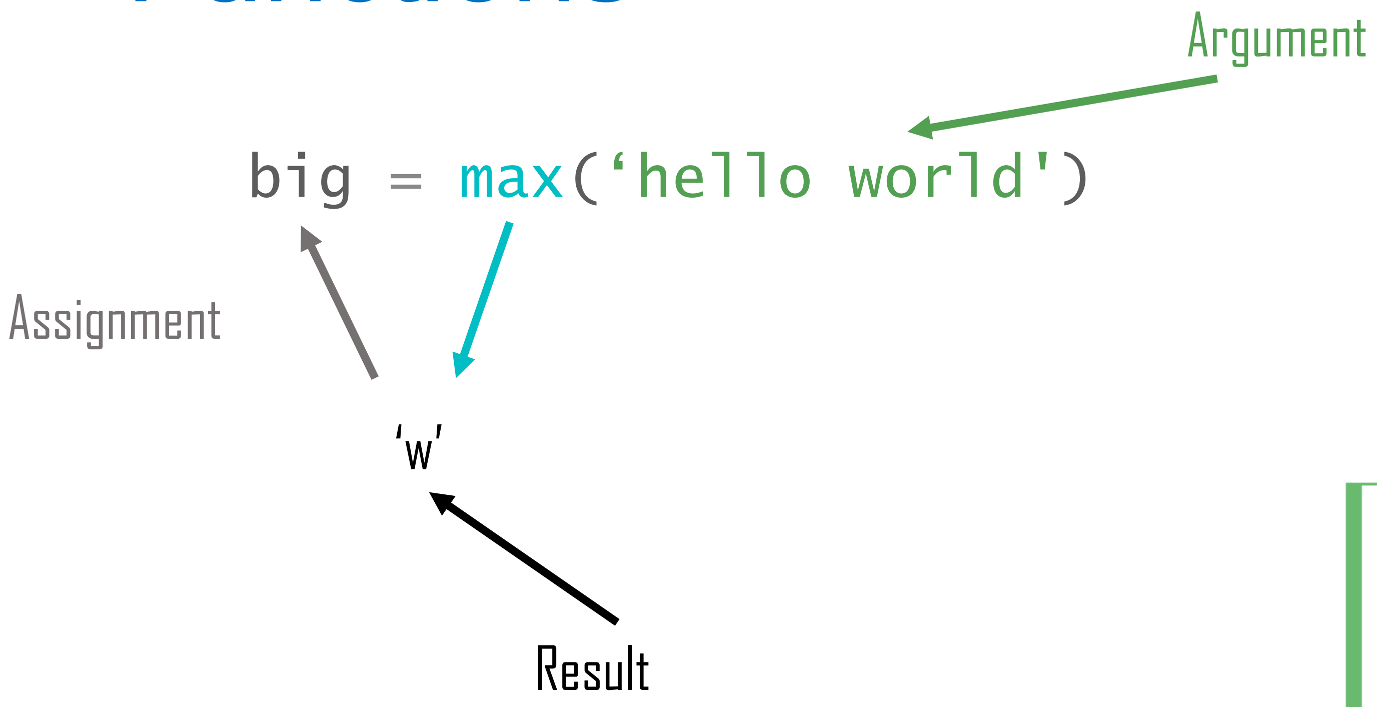
We call these reusable pieces of codes "functions"

4 functions.ipynb

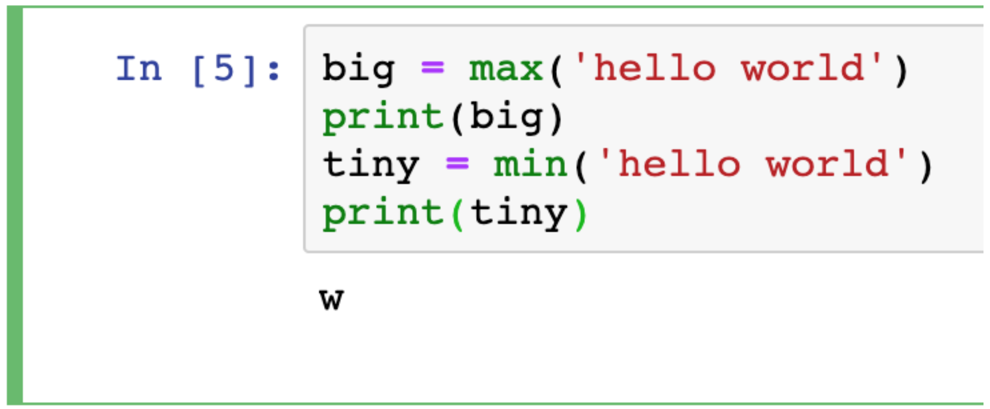
Functions

- A function is a named sequence of statements that performs a computation.
- When you define a function, you specify the name and the sequence of statements.
- Later, you can “call” the function by name.
- print(), input(), type(), float(), int(), etc.

Functions



```
big = max(‘hello world')
```



```
In [5]: big = max('hello world')
        print(big)
        tiny = min('hello world')
        print(tiny)

        w
```

4 functions.ipynb

Max Function

```
big = max('hello world')
print(big)
```

A function takes some `input` and produces an `output`

'hello world'
(a string)

→

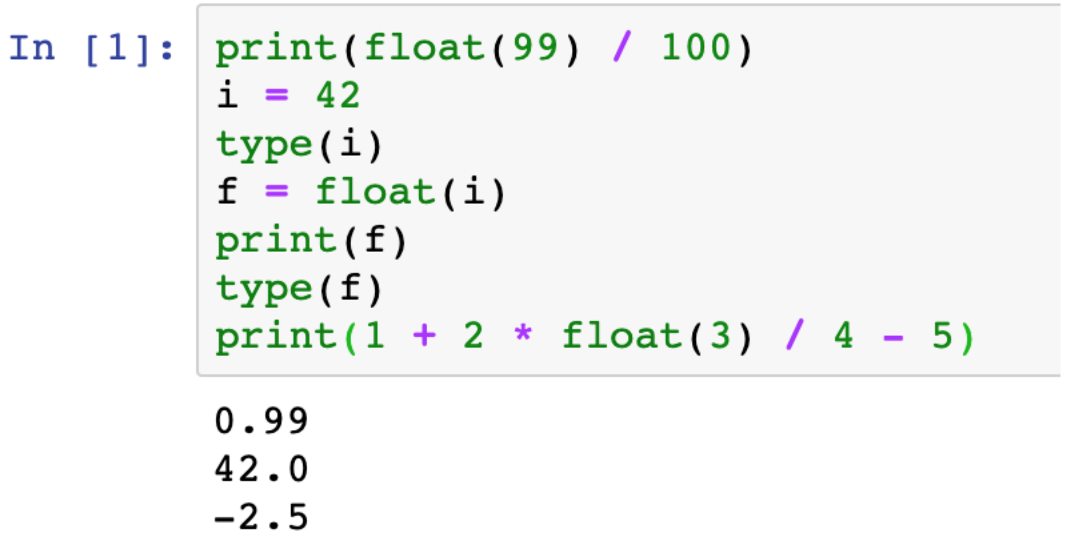
```
def max(inp):
    blah
    blah
    for x in inp:
        blah
        blah
```

→

'w'
(a string)

Type Conversions

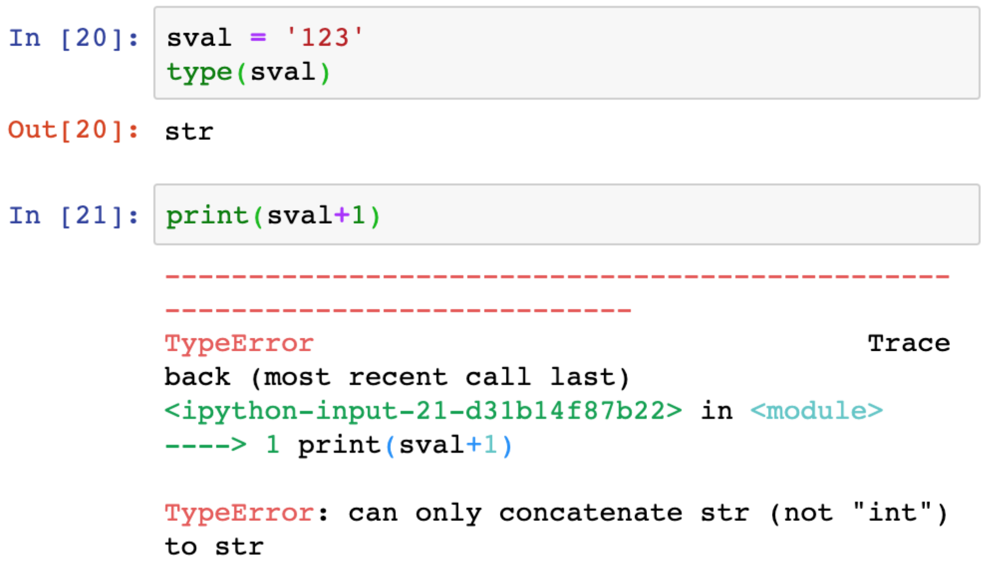
- When you put an integer and a floating point number in one expression, the integer is implicitly converted to a float
- You can control this with the built-in functions int() and float()



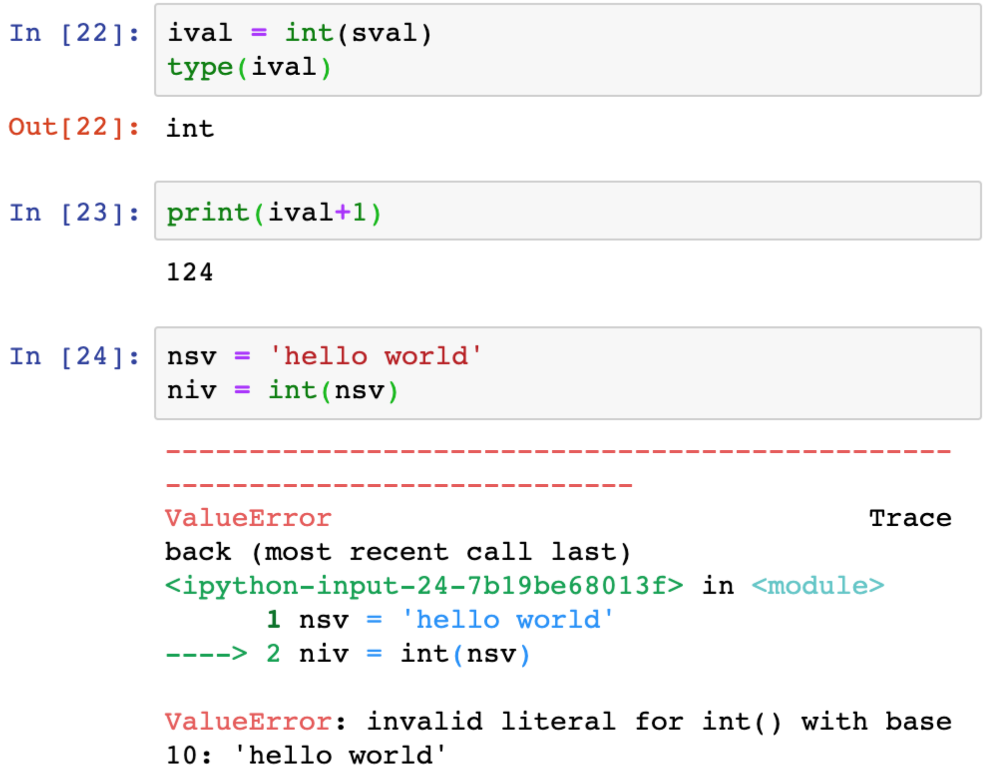
```
In [1]: print(float(99) / 100)
        i = 42
        type(i)
        f = float(i)
        print(f)
        type(f)
        print(1 + 2 * float(3) / 4 - 5)

        0.99
        42.0
        -2.5
```

String Conversions

- You can also use int() and float() to convert between strings and integers
- You will get an error if the string does not contain numeric characters



```
In [20]: sval = '123'
         type(sval)

Out[20]: str

In [21]: print(sval+1)

         ---------------------------------------------------
         ----------------------------
         TypeError                                    Trace
         back (most recent call last)
         <ipython-input-21-d31b14f87b22> in <module>
         ----> 1 print(sval+1)

         TypeError: can only concatenate str (not "int")
         to str
```



```
In [22]: ival = int(sval)
         type(ival)

Out[22]: int

In [23]: print(ival+1)

         124

In [24]: nsv = 'hello world'
         niv = int(nsv)

         ---------------------------------------------------
         ----------------------------
         ValueError                                   Trace
         back (most recent call last)
         <ipython-input-24-7b19be68013f> in <module>
               1 nsv = 'hello world'
         ----> 2 niv = int(nsv)

         ValueError: invalid literal for int() with base
         10: 'hello world'
```

Build Our Own Functions

- We create a new function using the def keyword followed by optional parameters in parentheses.
- We indent the body of the function
- This defines the function but does not execute the body of the function

```
def print_lyrics():
    print("I'm a lumberjack and I'm OK")
    print("I sleep all night and I work all day")
```

Build Our Own Functions

```
x = 5
print('Hello')

def print_lyrics():
    print("I'm a lumberjack and I'm OK")
    print("I sleep all night and I work all day")

print('Yo')
x = x + 2
print(x)
```

```
print_lyrics():
```

```
print("I'm a lumberjack and I'm
OK")
print("I sleep all night and I
work all day")
```

Output:

```
Hello
Yo
7
```

4 functions.ipynb

Build Our Own Functions

```
x = 5
print('Hello')

def print_lyrics():
   print("I'm a lumberjack and I'm OK")
   print("I sleep all night and I work all day")

print('Yo’)
print_lyrics()
x = x + 2
print(x)
```

Reuse/call/invoke

Output:

```
Hello
Yo
I'm a lumberjack and I'm OK
I sleep all night and I work all day
7
```

4 functions.ipynb

Arguments

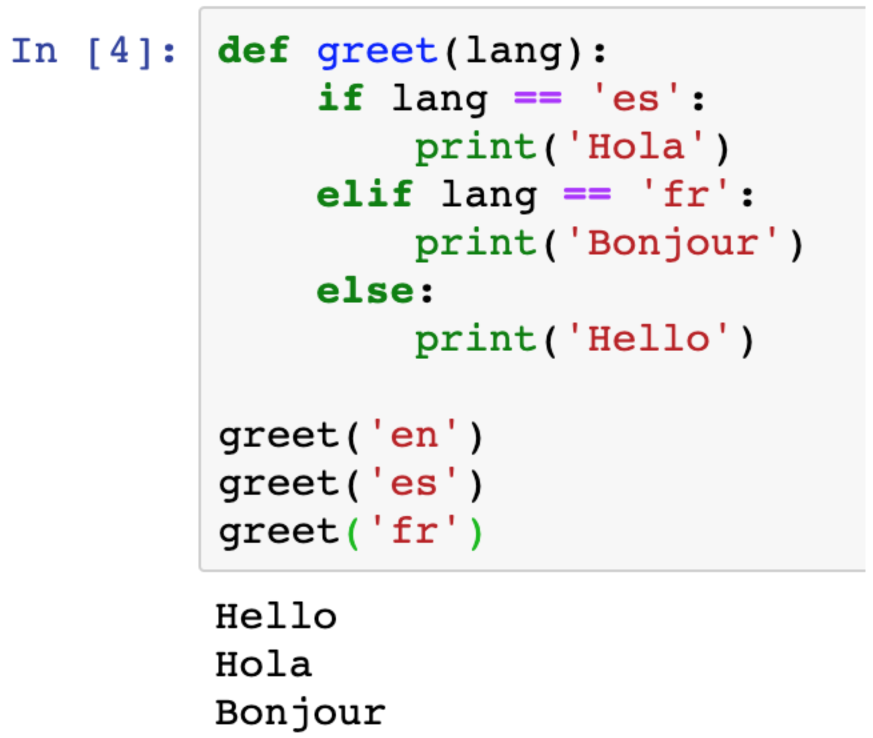
- An argument is a value we pass into the function as its input when we call the function
- We use arguments so that we can direct the function to do different kinds of work when we call it at different times
- We put the arguments in parentheses after name of the function

```
big = max('hello world')
```

Argument

Parameters

- A parameter is a variable which we use in the function definition. It is a "handle" that allows the code in the function to access the arguments for a particular function invocation.



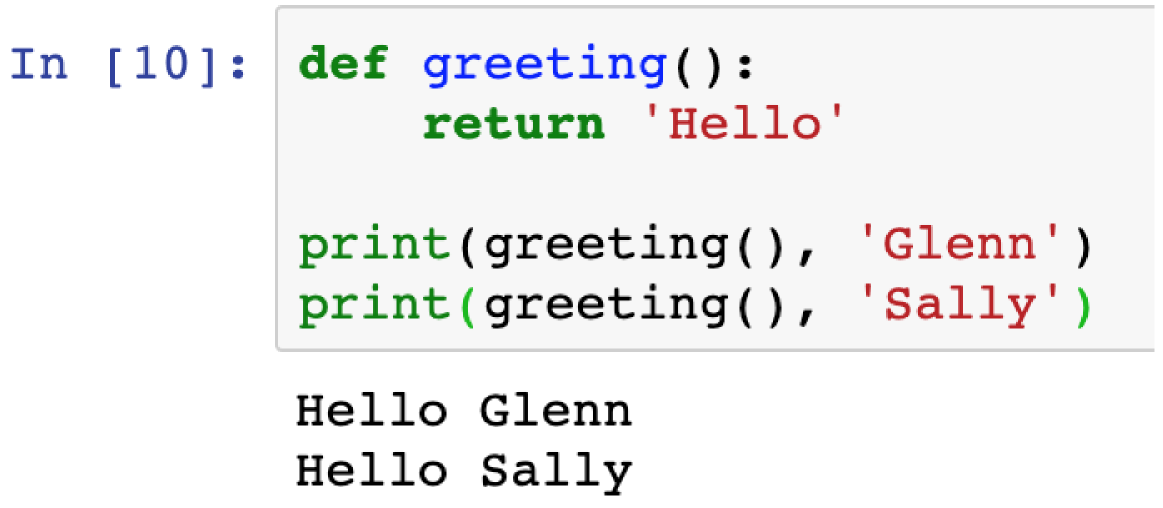
```
In [4]: def greet(lang):
            if lang == 'es':
                print('Hola')
            elif lang == 'fr':
                print('Bonjour')
            else:
                print('Hello')

        greet('en')
        greet('es')
        greet('fr')
```

```
Hello
Hola
Bonjour
```

4 functions.ipynb

Return Values

- Often a function will take its arguments, do some computation, and return a value to be used as the value of the function call in the calling expression.
- The return keyword is used for this.

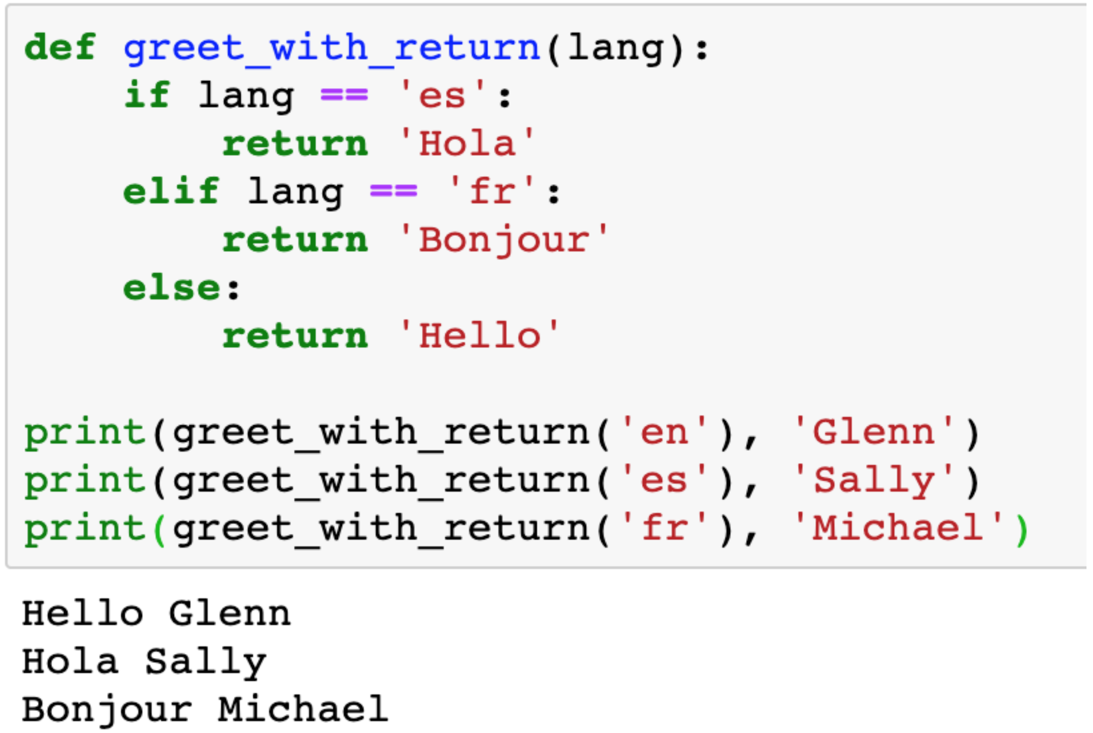


```
In [10]: def greeting():
             return 'Hello'

         print(greeting(), 'Glenn')
         print(greeting(), 'Sally')

         Hello Glenn
         Hello Sally
```

4 functions.ipynb

Return Values

- A "fruitful" function is one that produces a result (or return value)
- The return statement ends the function execution and "sends back" the result of the function.



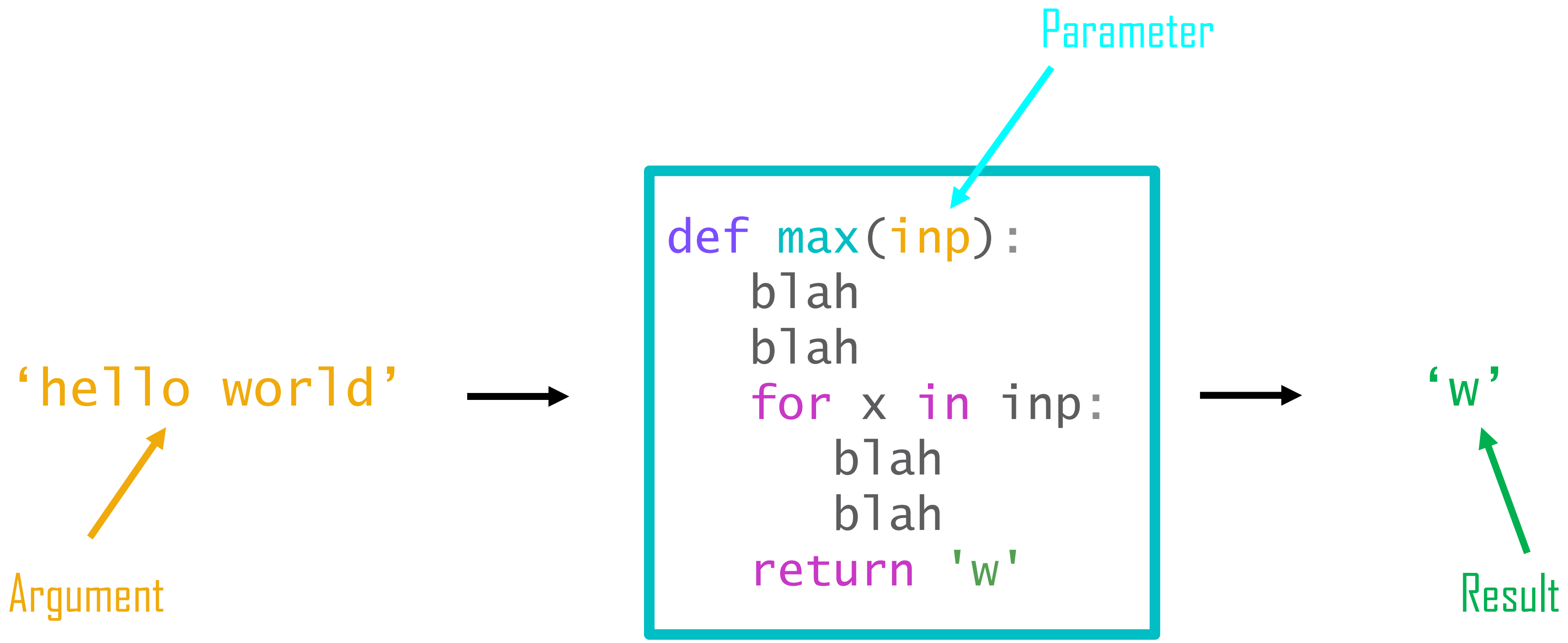
```python
def greet_with_return(lang):
    if lang == 'es':
        return 'Hola'
    elif lang == 'fr':
        return 'Bonjour'
    else:
        return 'Hello'

print(greet_with_return('en'), 'Glenn')
print(greet_with_return('es'), 'Sally')
print(greet_with_return('fr'), 'Michael')
```

```
Hello Glenn
Hola Sally
Bonjour Michael
```

4 functions.ipynb

Arguments, Parameter & Result

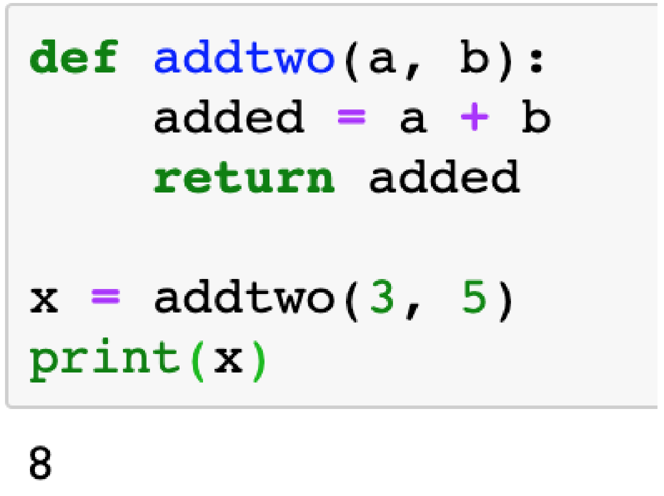
```
big = max('hello world')
print(big)
```



```
def max(inp):
    blah
    blah
    for x in inp:
        blah
        blah
    return 'w'
```

Multiple Parameters / Arguments

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters
- Some functions do not return values. We call them non-fruitful functions,
- If functions return values then we call them fruitful functions.



```python
def addtwo(a, b):
    added = a + b
    return added

x = addtwo(3, 5)
print(x)
```

```
8
```

4 functions.ipynb

Acknowledgements / Contributions

- Some of the slides used in this lecture from:
    - Charles R. Severance - University of Michigan School of Information

Thank You