Core Complete

Chris Piech and Mehran Sahami
CS106A, Stanford University

Learning Goals

1. Get more practice with variables and control flow
2. Understand information flow between functions
3. Learn about Python's doctest feature

Recall, Our Friend the Function

```
def main():
    avg = average(5.0, 10.2)
    print(avg)
```

function "call"

function "definition"

```
def average(a, b):
    sum = a + b
    return sum / 2
```

Recall, Our Friend the Function

arguments

```
def main():
    avg = average(5.0, 10.2)
    print(avg)
```

parameters

```
def average(a, b):
    sum = a + b
    return sum / 2
```

return value

Parameters

Parameters let you provide a function with some information when you are calling it.

Returns

Returns let you receive information from a function

No Functions in Functions

```
def main():
    print("hello world")
    def say_goodbye():
        print("goodbye!")
```

Technically legal, but often a sign at the start that you are confusing function *definition* and function *call*

No Functions in Functions

```
def main():
    print("hello world")
    say_goodbye()


def say_goodbye():
    print("goodbye!")
```


```
def get_sentiment(user_text)
```

Factorial

`factorial(n)`

Examples:

```
factorial(5) = 5 * 4 * 3 * 2 * 1

factorial(3) = 3 * 2 * 1

factorial(1) = 1

factorial(0) = 1
```

Understand the mechanism

Tracing a Program

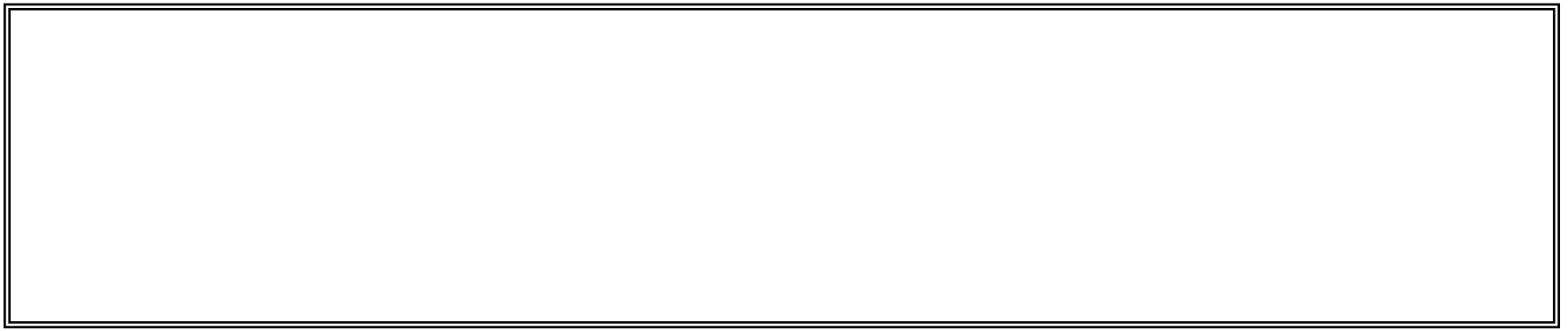
```
# Constant – visible to all functions
MAX_NUM = 4

def main():
    # repeat for several values!
    for i in range(MAX_NUM):
        print(i, factorial(i))

def factorial(n):
    """
    Calculates n factorial.
    5 factorial is 5 * 4 * 3 * 2 * 1
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Tracing a Program

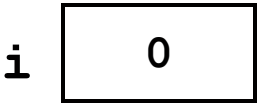
```
# Constant – visible to all functions
MAX_NUM = 4

def main():
    # repeat for several values!
    for i in range(MAX_NUM):
        print(i, factorial(i))

def factorial(n):
    """
    Calculates n factorial.
    5 factorial is 5 * 4 * 3 * 2 * 1
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

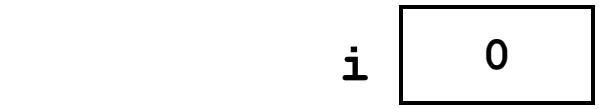
i 



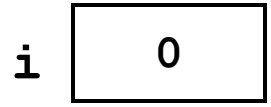
```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```



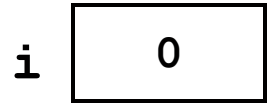
```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```



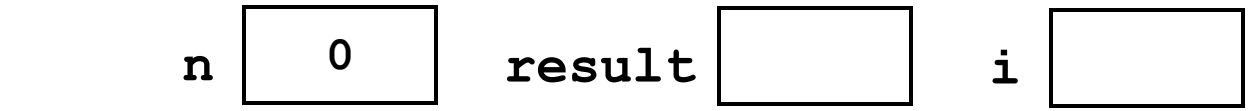
```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```



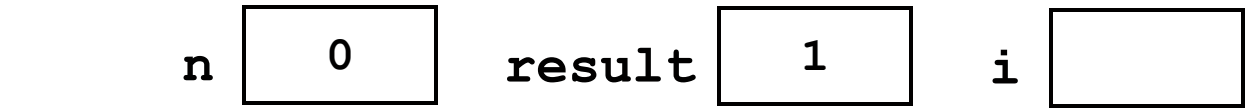
```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```



```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```



```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```



```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

n	0	result	1	i	1

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[]

n	0	result	1	i	1

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

n 0 result 1 i 1

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

1

i 0

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

1

i | 0

```
0   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

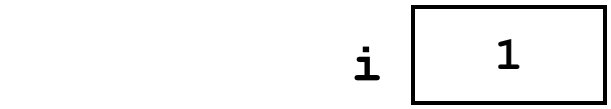

```
0  1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 1

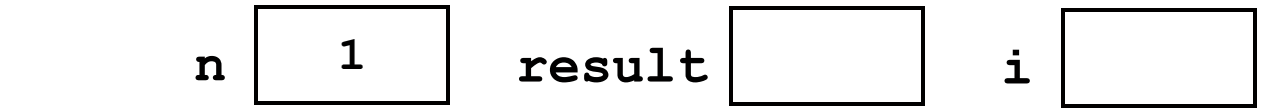
```
0  1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```



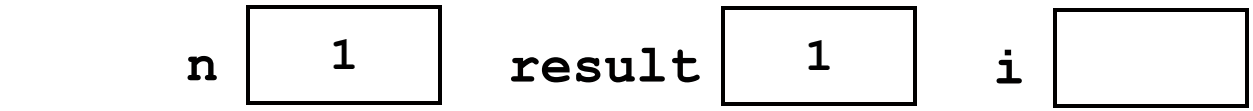

```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```



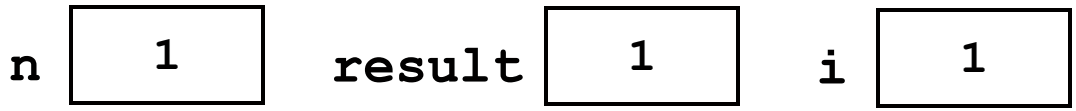

```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```




```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```




```
0  1
```

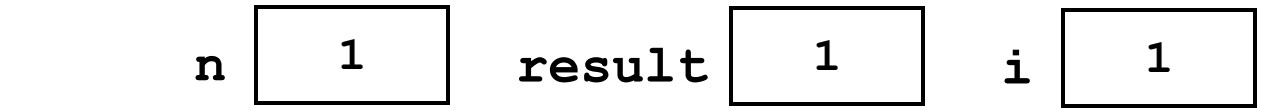
```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1]

n	1	result	1	i	1

```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```




```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1]

n	1	result	1	i	2

```
0  1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

n 1 result 1 i 2

```
0  1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

1

i 1

```
0  1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

1

i 1

```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 2

```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 2

```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 2

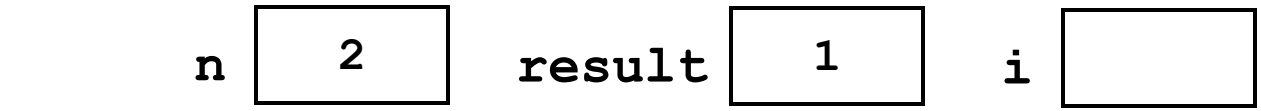
```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 2

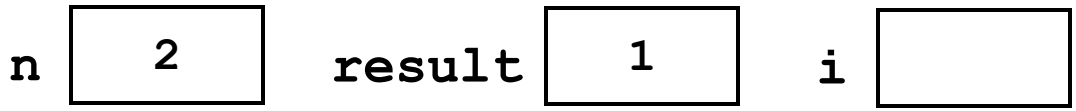
```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```




```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```




```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1,2]

n	2	result	1	i	

```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1,2]

n	2	result	1	i	1

```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1,2]

n	2	result	1	i	2

```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1,2]

n	2	result	2	i	2

```
0   1
1   1
```

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

[1,2]

n	2	result	2	i	2

```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

2

i 2

```
0   1
1   1
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

2

i 2

```
0   1
1   1
2   2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i 3

```
0  1
1  1
2  2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 3

```
0   1
1   1
2   2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i 3

```
0  1
1  1
2  2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i 3

```
0  1
1  1
2  2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

6

i 3

```
0   1
1   1
2   2
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

6

i 3

```
0   1
1   1
2   2
3   6
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 4

```
0   1
1   1
2   2
3   6
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 4

```
0   1
1   1
2   2
3   6
```

```
def main():
    for i in range(MAX_NUM):
        print(i, factorial(i))
```

i | 4

Done!

```
0   1
1   1
2   2
3   6
```

Parameters

Every time a function is called, new parameters are created, initially bound to the values given.

An interlude: testing

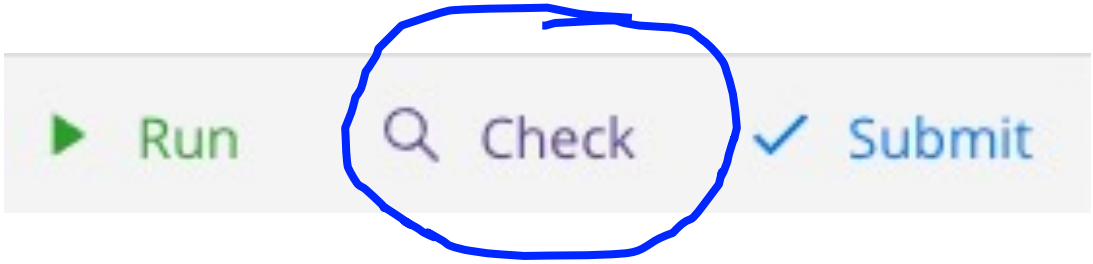
Testing: *Why* doesn't your program work?

Doctest

```
def factorial(n):
    """
    This function returns the factorial of n
    Input: n (number to compute the factorial of)
    Returns: value of n factorial
    Doctests:
    >>> factorial(3)
    6
    >>> factorial(1)
    1
    >>> factorial(0)
    1
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Doctest

```
def factorial(n):
    """
    This function returns the factorial of n
    Input: n (number to compute the factorial of)
    Returns: value of n factorial
    Doctests:
    >>> factorial(3)
    6
    >>> factorial(1)
    1
    >>> factorial(0)
    1
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```




```
# or, in pycharm
python -m doctest fact.py -v
```

When you need to know the step by step

Bad Times With functions

```
# NOTE: This program is buggy!!

def add_five(x):
    x += 5



def main():
    x = 3
    add_five(x)
    print("x = " + str(x))
```

Let's "trace" this program

Bad Times With functions

```
# NOTE: This program is buggy!!

def add_five(x):
    x += 5



def main():
    x = 3
    add_five(x)
    print("x = " + str(x))
```

Good Times With functions

```
# NOTE: This program is feeling just fine...

def add_five(x):
    x += 5
    return x


def main():
    x = 3
    x = add_five(x)
    print("x = " + str(x))
```


Parameter passing mechanism

If you reassign a variable inside a helper function, and you want the change to persist use the pattern:

```
x = change(x)
```

Good Times With functions

```
# NOTE: This program is feeling just fine...

def add_five(x):
    x += 5
    return x


def main():
    x = 3
    x = add_five(x)
    print("x = " + str(x))
```

When we want to "reassign" x inside a helper function, employ the `x = change(x)` pattern!

No inherent connection between these two

```
# NOTE: This program is buggy!!

def add_five(x):
    x += 5

def main():
    x = 3
    add_five(x)
    print("x = " + str(x))
```

These are two separate variables. They are not linked!

Only relationship: value of main's x is used when creating add_five's x

Parameter passing mechanism

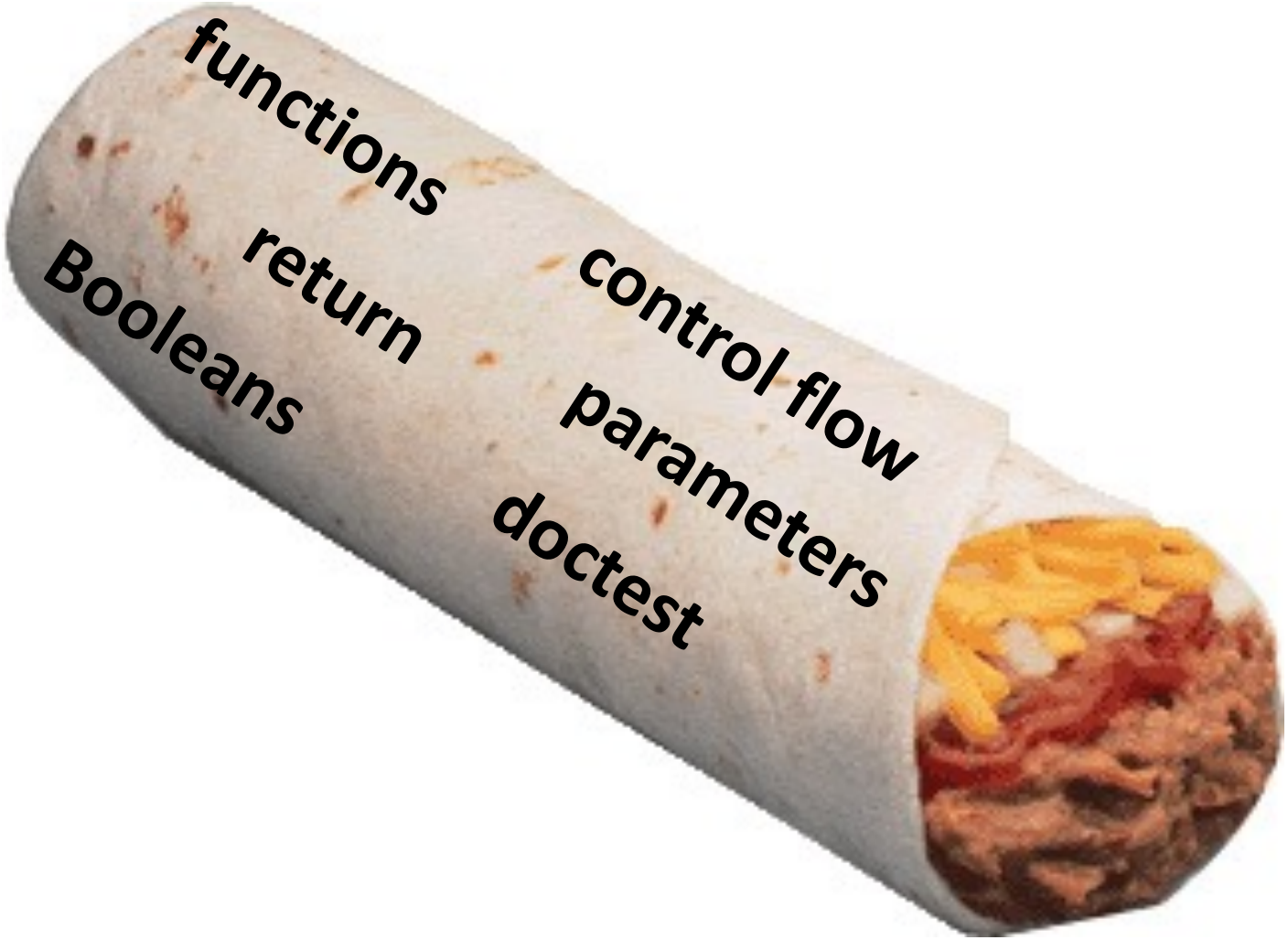
When a parameter is passed during a function call, a **new variable** is created for the lifetime of the function call.

That new variable may or may not have the **same name** as the value that was passed in!

Later on in class… we will see cases where changes to variables in helper functions seem to persist! It will be great. We will let you know when we get there and exactly when that happens!

Learning Goals

1. Get more practice with function parameters
2. Understand information flow in a program
3. Learn about Python's doctest feature


functions
return
Booleans
control flow
parameters
doctest

Console Terminal

```
Welcome to Rock Paper Scissors
You will play 3 games against the AI
rock beats scissors
scissors beats paper
paper beats rock
-----------------------------------------

what do you play? rock
The AI plays: scissors
The winner is: human

what do you play? paper
The AI plays: scissors
The winner is: ai

what do you play? rock
The AI plays: paper
The winner is: ai

your score -1
```