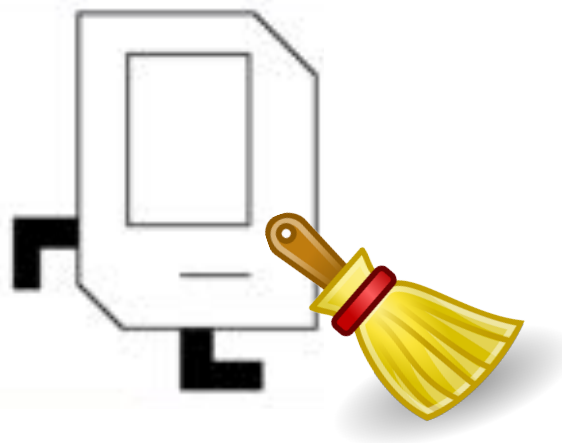# Functions and Parameters
Chris Piech and Mehran Sahami
CS106A, Stanford University

Housekeeping

- Reminder: Diagnostic is on Monday
  - Takes place during class time
  - Like an exam
    - Really meant for you to gauge your understanding
  - Covers material through today
  - Email Brahm if you have a time conflict or are outside the Americas

Learning Goals

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

Recall, Our Friend the Function

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

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
```

Parameters

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

A Full Program

```python
# Constant - visible to all functions
MAX_NUM = 4


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
```

A Full Program

```python
# Constant - visible to all functions
MAX_NUM = 4


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
```

Understand the mechanism

```
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))
```

i   0

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

i   [0]

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

i | 0

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

i | 0

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

    return result
```

n `0`   result `    `   i `    `

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

    return result
```

n `0`   result `1`   i

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

    return result
```

n 0   result 1   i 1

```python
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))
```

i `1`

```
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))
```

i   1

0   1

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

    return result
```

n `1`   result `    `   i `    `

0   1

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

    return result
```

n [1]   result [1]   i [ ]

0   1

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

    return result
```

n `1`   result `1`   i `1`

0   1

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

    return result
```

n `1`   result `1`   i `1`

0 1

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

    return result
```

n `1`   result `1`   i `1`

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 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 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 memory is created for that call.

Parameter values are passed in.

All *local* variables start fresh (no old values)

An interlude:
doctest

Doctest

```python
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

```python
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
```

Say this was in file "fact.py"
To run doctests (on PC):

```
> py -m doctest fact.py -v
```

Let's try it!!

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))
```

Bad Times With functions

```python
# 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

```python
# 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))
```

For primitive types (e.g., int, float, Boolean, string):

- **Copies** of values are passed as parameters.

- Variable that was passed in as an argument is **not** changed when you modify parameter in the function.

Pass by "Value"

Careful!

No Functions in Functions

```python
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

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


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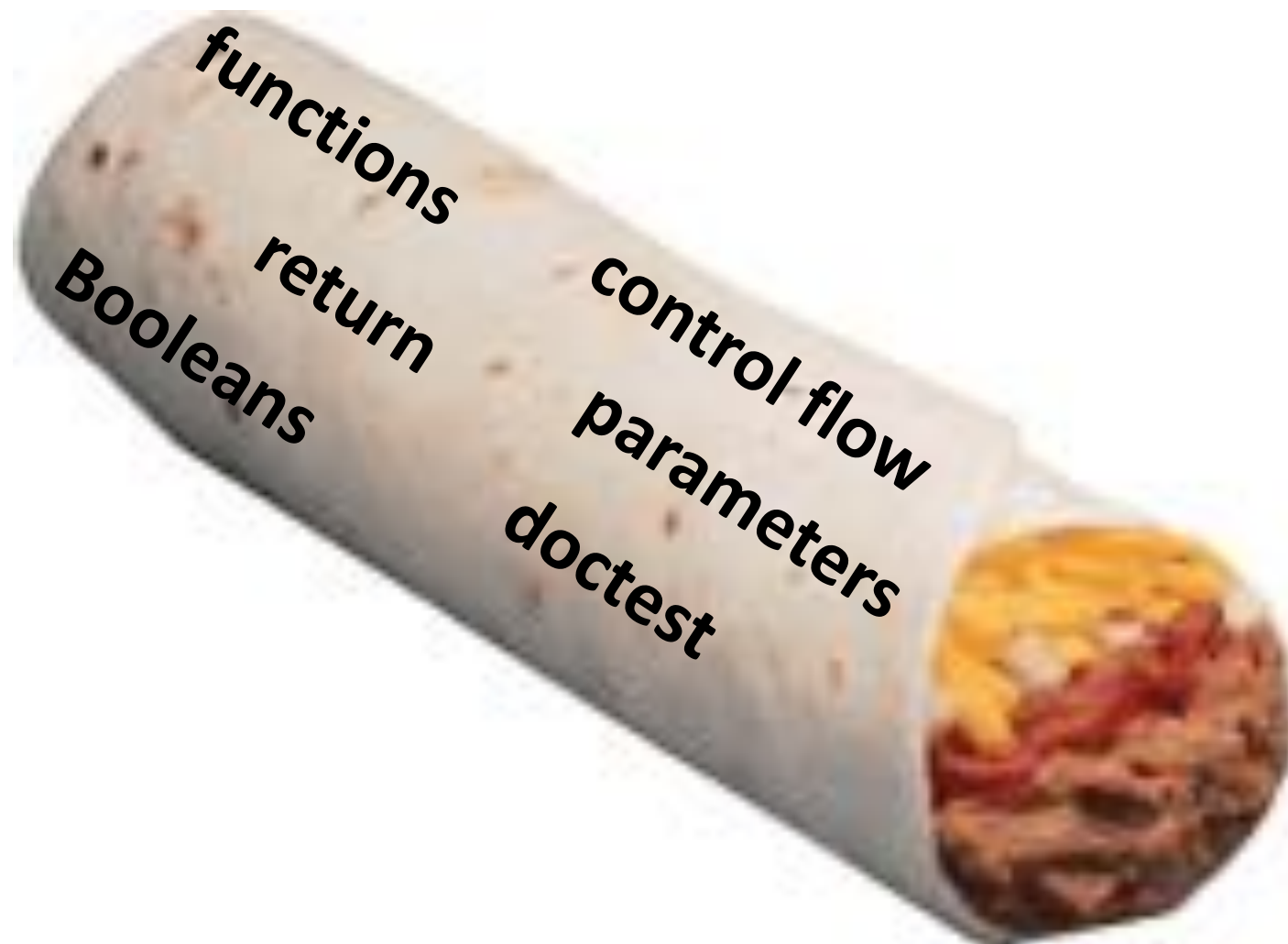
Learning Goals

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

The Whole Burrito:
calendar.py