

Function Examples

Class outline:

- *Args
- Currying
- Midterm Review

*Args

The *args syntax

What if you want a function to accept any number of arguments?

The built-in `max` function allows that:

```
max(1, 2) # 2
max(10, 30, 20) # 30
max(-2, 33, -40, 400, 321) # 400
```

The `*args` syntax

What if you want a function to accept any number of arguments?

The built-in `max` function allows that:

```
max(1, 2) # 2
max(10, 30, 20) # 30
max(-2, 33, -40, 400, 321) # 400
```

That's possible by using the `*args` syntax in the function definition.

```
def max(*args):
    # Do something with *args
```

Forwarding the *args

One way to use `*args` is to send those arguments into another function.

```
def min_and_max(*args):  
    return min(*args), max(*args)
```

```
min_and_max(-2, 33, -40, 400, 321)
```

Forwarding the *args

One way to use `*args` is to send those arguments into another function.

```
def min_and_max(*args):  
    return min(*args), max(*args)
```

```
min_and_max(-2, 33, -40, 400, 321) # -40, 400
```

Forwarding HOF example

A HOF can return a function that can be called with any number of arguments, and then forward those arguments inside the returned function.

```
def printed(f):  
    def print_and_return(*args):  
        result = f(*args)  
        print('Result:', result)  
        return result  
    return print_and_return
```

```
printed_max = printed(max)  
printed_max(-2, 33, -40, 400, 321)
```


Currying

(Reminder) Function currying

Currying: Converting a function that takes multiple arguments into a single-argument higher-order function.

A function that currys any two-argument function:

```
def curry2(f):  
    def g(x):  
        def h(y):  
            return f(x, y)  
        return h  
    return g
```

```
from operator import add  
  
make_adder = curry2(add)  
make_adder(2)(3)
```

```
curry2 = lambda f: lambda x: lambda y: f(x, y)
```

Use case for currying #1

Whenever another function requires a function that only takes one argument:

```
def transform_numbers(num1, num2, num3, transform):  
    return transform(num1), transform(num2), transform(num3)
```

```
transform_numbers(3, 4, 5, curry2(add)(60))
```

Use case for currying #1

Whenever another function requires a function that only takes one argument:

```
def transform_numbers(num1, num2, num3, transform):  
    return transform(num1), transform(num2), transform(num3)
```

```
transform_numbers(3, 4, 5, curry2(add)(60))
```

Alternate approach:

```
transform_numbers(3, 4, 5, lambda x: add(60, x))
```

Use case for currying #2

Turning a generalized function into a specialized function:

```
def html_tag(tag_name, text):  
    return "<" + tag_name + ">" + text + "</" + tag_name + ">"  
  
p_tag = curry2(html_tag)("p")  
p_tag("hello hello")
```

Use case for currying #2

Turning a generalized function into a specialized function:

```
def html_tag(tag_name, text):  
    return "<" + tag_name + ">" + text + "</" + tag_name + ">"  
  
p_tag = curry2(html_tag)("p")  
p_tag("hello hello")
```

Alternate approach:

```
import functools  
  
p_tag = functools.partial(html_tag, "p")  
p_tag("hello hello")
```

Why learn currying in Python?

It's good for you!

CS61A introduces many concepts that aren't standard Python practice, but that show up in other languages.

Currying is a very common practice in functional programming languages like Haskell or Clojure.

Review

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|-----------------------|---------------------|---------------------------|
|-----------------------|---------------------|---------------------------|

| | | |
|----------------|--|--|
| <code>5</code> | | |
|----------------|--|--|

| | | |
|-----------------------|--|--|
| <code>print(5)</code> | | |
|-----------------------|--|--|

| | | |
|------------------------------|--|--|
| <code>print(print(5))</code> | | |
|------------------------------|--|--|

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | |
| <code>print(5)</code> | | |
| <code>print(print(5))</code> | | |

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | | |
| <code>print(print(5))</code> | | |

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | | |
| <code>print(print(5))</code> | | |

```
>> 5
5
```

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | <code>None</code> | |
| <code>print(print(5))</code> | | |

```
>> 5
5
```

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | <code>None</code> | <code>5</code> |
| <code>print(print(5))</code> | | |

```
>> 5
5
```

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | <code>None</code> | <code>5</code> |
| <code>print(print(5))</code> | | |

```
>> 5
5
>>> print(5)
5
```

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|---------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | <code>None</code> | <code>5</code> |
| <code>print(print(5))</code> | <code>None</code> | |

```
>> 5
5
>>> print(5)
5
```


What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|------------------------------|---------------------|-------------------------------------|
| <code>5</code> | <code>5</code> | <code>5</code> |
| <code>print(5)</code> | <code>None</code> | <code>5</code> |
| <code>print(print(5))</code> | <code>None</code> | <code>5</code> <code>None</code> |

```
>> 5
5
>>> print(5)
5
```

What Would Python Do? #1

WWPD exercises test our understanding of how Python evaluates code and what it chooses to display in the shell.

| The expression | Evaluates to | Interactive output |
|-----------------------|---------------------|---------------------------|
| 5 | 5 | 5 |
| print(5) | None | 5 |
| print(print(5)) | None | 5 None |

```
>> 5
5
>>> print(5)
5
>>> print(print(5))
5
None
```

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

The expression

Evaluates to

Interactive output

```
delay(6)()
```

```
delay(delay)()(6)()
```

```
print(delay(print)()(4))
```

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

The expression

Evaluates to

Interactive output

`delay(6)()`

`6`

`delay(delay)()(6)()`

`print(delay(print)()(4))`

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

The expression

Evaluates to

Interactive output

`delay(6)()`

`6`

`delayed`
`6`

`delay(delay)()(6)()`

`print(delay(print)()(4))`

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

| The expression | Evaluates to | Interactive output |
|---------------------------------------|--------------|--------------------|
| <code>delay(6)()</code> | 6 | delayed 6 |
| <code>delay(delay)()(6)()</code> | 6 | |
| <code>print(delay(print)()(4))</code> | | |

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

| The expression | Evaluates to | Interactive output |
|---------------------------------------|--------------|-------------------------|
| <code>delay(6)()</code> | 6 | delayed 6 |
| <code>delay(delay)()(6)()</code> | 6 | delayed delayed 6 |
| <code>print(delay(print)()(4))</code> | | |

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

| The expression | Evaluates to | Interactive output |
|---------------------------------------|-------------------|--|
| <code>delay(6)()</code> | <code>6</code> | <code>delayed</code> <code>6</code> |
| <code>delay(delay)()(6)()</code> | <code>6</code> | <code>delayed</code> <code>delayed</code> <code>6</code> |
| <code>print(delay(print)()(4))</code> | <code>None</code> | |

What Would Python Do? #2

```
def delay(arg):  
    print('delayed')  
    def g():  
        return arg  
    return g
```

| The expression | Evaluates to | Interactive output |
|---------------------------------------|-------------------|--|
| <code>delay(6)()</code> | <code>6</code> | <code>delayed</code> <code>6</code> |
| <code>delay(delay)()(6)()</code> | <code>6</code> | <code>delayed</code> <code>delayed</code> <code>6</code> |
| <code>print(delay(print)()(4))</code> | <code>None</code> | <code>delayed</code> <code>4</code> <code>None</code> |

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

The expression

Evaluates to

Interactive output

```
pirate('treasure')('scurvy')
```

```
add(pirate(3)(square)(4), 1)
```

```
pirate(pirate(pirate))(5)(7)
```

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

The expression

Evaluates to

Interactive output

```
pirate('treasure')('scurvy')
```

```
'scurvy'
```

```
add(pirate(3)(square)(4), 1)
```

```
pirate(pirate(pirate))(5)(7)
```

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

The expression

Evaluates to

Interactive output

```
pirate('treasure')('scurvy')
```

```
'scurvy'
```

```
matey
```

```
'scurvy'
```

```
add(pirate(3)(square)(4), 1)
```

```
pirate(pirate(pirate))(5)(7)
```

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

| The expression | Evaluates to | Interactive output |
|---|-----------------------|---|
| <code>pirate('treasure')('scurvy')</code> | <code>'scurvy'</code> | <code>matey</code> <code>'scurvy'</code> |
| <code>add(pirate(3)(square)(4), 1)</code> | <code>17</code> | |
| <code>pirate(pirate(pirate))(5)(7)</code> | | |

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

| The expression | Evaluates to | Interactive output |
|---|--------------|--------------------|
| <code>pirate('treasure')('scurvy')</code> | 'scurvy' | matey 'scurvy' |
| <code>add(pirate(3)(square)(4), 1)</code> | 17 | matey 17 |
| <code>pirate(pirate(pirate))(5)(7)</code> | | |

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

| The expression | Evaluates to | Interactive output |
|---|--------------|--------------------|
| <code>pirate('treasure')('scurvy')</code> | 'scurvy' | matey 'scurvy' |
| <code>add(pirate(3)(square)(4), 1)</code> | 17 | matey 17 |
| <code>pirate(pirate(pirate))(5)(7)</code> | Error | |

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

What Would Python Do? #3

```
def pirate(arggg):  
    print('matey')  
    def plunder(arggg):  
        return arggg  
    return plunder
```

| The expression | Evaluates to | Interactive output |
|---|--------------|-------------------------|
| <code>pirate('treasure')('scurvy')</code> | 'scurvy' | matey 'scurvy' |
| <code>add(pirate(3)(square)(4), 1)</code> | 17 | matey 17 |
| <code>pirate(pirate(pirate))(5)(7)</code> | Error | matey matey Error |

A name evaluates to the value bound to that name in the earliest frame of the current environment in which that name is found.

Environment Diagram

```
def horse(mask):  
    horse = mask  
    def mask(horse):  
        return horse  
    return horse(mask)
```

```
mask = lambda horse: horse(2)  
horse(mask)
```

Global frame

| | |
|-------|--|
| horse | |
| mask | |

f1:

| | |
|--------------|--|
| | |
| | |
| Return value | |

f2:

,

| | |
|--------------|--|
| | |
| | |
| Return value | |

f3:

| | |
|--------------|--|
| | |
| | |
| Return value | |

Implementing a function

```
def remove(n, digit):  
    """Return digits of non-negative N  
    that are not DIGIT, for some  
    non-negative DIGIT less than 10.  
    >>> remove(231, 3)  
    21  
    >>> remove(243132, 2)  
    4313  
    """  
    kept = 0  
    digits = 0  
    while _____:  
        last = n % 10  
        n = n // 10  
        if _____:  
            kept = _____  
            digits = _____  
    return _____
```

Implementing a function

```
def remove(n, digit):  
    """Return digits of non-negative N  
    that are not DIGIT, for some  
    non-negative DIGIT less than 10.  
    >>> remove(231, 3)  
    21  
    >>> remove(243132, 2)  
    4313  
    """  
    kept = 0  
    digits = 0  
    while _____:  
        last = n % 10  
        n = n // 10  
        if _____:  
            kept = _____  
            digits = _____  
    return _____
```

- Read the description

Implementing a function

```
def remove(n, digit):  
    """Return digits of non-negative N  
    that are not DIGIT, for some  
    non-negative DIGIT less than 10.  
    >>> remove(231, 3)  
    21  
    >>> remove(243132, 2)  
    4313  
    """  
    kept = 0  
    digits = 0  
    while _____:  
        last = n % 10  
        n = n // 10  
        if _____:  
            kept = _____  
            digits = _____  
    return _____
```

- Read the description
- Verify the examples & pick a simple one

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while _____:
        last = n % 10
        n = n // 10
        if _____:
            kept = _____
            digits = _____
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while _____:
        last = n % 10
        n = n // 10
        if _____:
            kept = _____
            digits = _____
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template
- Implement without the template, then change your implementation to match the template.
OR If the template is helpful, use it.

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while _____:
        last = n % 10
        n = n // 10
        if _____:
            kept = _____
            digits = _____
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template
- Implement without the template, then change your implementation to match the template.
OR If the template is helpful, use it.
- Annotate names with values from your chosen example

Implementing a function

```
def remove(n, digit):  
    """Return digits of non-negative N  
    that are not DIGIT, for some  
    non-negative DIGIT less than 10.  
    >>> remove(231, 3)  
    21  
    >>> remove(243132, 2)  
    4313  
    """  
    kept = 0  
    digits = 0  
    while _____:  
        last = n % 10  
        n = n // 10  
        if _____:  
            kept = _____  
            digits = _____  
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template
- Implement without the template, then change your implementation to match the template.
OR If the template is helpful, use it.
- Annotate names with values from your chosen example
- Write code to compute the result

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while _____:
        last = n % 10
        n = n // 10
        if _____:
            kept = _____
            digits = _____
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template
- Implement without the template, then change your implementation to match the template.
OR If the template is helpful, use it.
- Annotate names with values from your chosen example
- Write code to compute the result
- Did you really return the right thing?

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while _____:
        last = n % 10
        n = n // 10
        if _____:
            kept = _____
            digits = _____
    return _____
```

- Read the description
- Verify the examples & pick a simple one
- Read the template
- Implement without the template, then change your implementation to match the template.
OR If the template is helpful, use it.
- Annotate names with values from your chosen example
- Write code to compute the result
- Did you really return the right thing?
- Check your solution with the other examples

Implementing a function

```
def remove(n, digit):
    """Return digits of non-negative N
    that are not DIGIT, for some
    non-negative DIGIT less than 10.
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept = 0
    digits = 0
    while n > 0:
        last = n % 10
        n = n // 10
        if last != digit:
            kept = kept + (last * 10 ** digits)
            digits = digits + 1
    return kept
```