CIS 122

Midterm and Onwards

Logistics

- Midterm is graded
  - Haven't recorded grades yet
  - Might curve somewhat
  - Give midterms back at end of class!
- Assignment 3 not graded yet
  - Hopefully by class tomorrow
- Assignment 4 will be posted soon
  - Hopefully by class tomorrow
  - Due Sunday at midnight

Part 1

- Lots of evaluations
- Generally correct

Part 2

```
a = 3
b = 5

if a < 5:
    a = a + 5
elif b < 10:
    a = a + 10
else:
    a = b

a = a + b
b = a + b
```

Part 2

```
a = 3
b = 5

if a < 5:
    a = a + 5
elif b < 10:
    a = a + 10
else:
    a = b

a = a + b
b = a + b
```

a = 3
b = 5

Part 2

```
a = 3
b = 5

if a < 5:
    a = a + 5
elif b < 10:
    a = a + 10
else:
    a = b

a = a + b
b = a + b
```

a = 3
b = 5

a = 8
b = 5

Part 2

```
a = 3
b = 5

if a < 5:
    a = a + 5
elif b < 10:
    a = a + 10
else:
    a = b

a = a + b
b = a + b
```

a = 3
b = 5

a = 8
b = 5

a = 13
b = 18

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

Part 3

__main__

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

n	→ 5
s	→ 'abcdef'
string1	→ swap('abcdef')

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

n	→ 5
s	→ 'abcdef'
string1	→ swap('abcdef')

swap

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

n	→ 5
s	→ 'abcdef'
string1	→ swap('abcdef')

swap

string	→ 'abcdef'
half	→ 3
first	→ 'abc'
rest	→ 'def'
swapped	→ 'defabc'

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

n	→ 5
s	→ 'abcdef'
string1	→ 'defabc'

swap

string	→ 'abcdef'
half	→ 3
first	→ 'abc'
rest	→ 'def'
swapped	→ 'defabc'

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ mystery(5, 'abcdef')

mystery

n	→ 5
s	→ 'abcdef'
string1	→ 'defabc'
string2	→ 'defabcdef...'

swap

string	→ 'abcdef'
half	→ 3
first	→ 'abc'
rest	→ 'def'
swapped	→ 'defabc'

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ 'defabcdef...'

mystery

n	→ 5
s	→ 'abcdef'
string1	→ 'defabc'
string2	→ 'defabcdef...'

swap

string	→ 'abcdef'
half	→ 3
first	→ 'abc'
rest	→ 'def'
swapped	→ 'defabc'

Part 3

```
def swap(string):
    half = len(string)/2
    first = string[:half]
    rest = string[half:]
    swapped = rest+first
    return swapped

def mystery(n,s):
    string1 = swap(s)
    string2 = n * string1
    return string2

x = 5
y = 'abcdef'
z = mystery(x,y)
```

__main__

swap	→ <func>
mystery	→ <func>
x	→ 5
y	→ 'abcdef'
z	→ 'defabcdef...'

mystery

n	→ 5
s	→ 'abcdef'
string1	→ 'defabc'
string2	→ 'defabcdef...'

swap

string	→ 'abcdef'
half	→ 3
first	→ 'abc'
rest	→ 'def'
swapped	→ 'defabc'

Part 4

```
def something(a,b):
    '''What do I do?'''

    c = a-b
    if c==0:
        return True
    else:
        return False
```

Part 5

```
def function(string, num):
    '''What do I do?'''

    if string == '':
        print 'I can't do that!'
    elif num == 0:
        return string[0]
    else:
        return function(string[1:], num-1)
```

Part 6

F0 = 0 $\quad\quad$ fib(0) → 0

F1 = 1 $\quad\quad$ fib(1) → 1

Fn = F(n-1) + F(n-2) $\quad\quad$ fib(n) → fib(n-1) + fib(n-2)

```
def fibonacci(n):
    '''Computes the nth fibonacci number'''
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```