CIS 122

Midterm and Onwards

Grading Status

- Homework 3 isn't graded yet
  - And might not be for a few days...

- But your midterms are graded
  - Generally impressed

How did you do?

```
def gradeChecker(x):
    if x > 90:
        print "Awesome"
    elif x > 80:
        print "Good Job"
    elif x > 70:
        print "Might want to do a little review"
    else:
        print "Come talk to me"
```

Part 1

- 2 + 3 * 4
    - Order of Operations
    - Multiplication before addition

- 10 / 4
    - Integer Division

- 10 * 10.0
    - Float Multiplication

- 10.0 / 4
    - Float Division

Part 1

- "Beseech"[5]
    - Count from the left
    - Start from 0
- "Beseech"[-2]
    - Count from the right
    - Start from -1
- "Beseech"[2:5]
    - Start from "Beseech"[2]
    - Up to (but not including) "Beseech"[5]

Part 2

- not True or False
  - How does Python parse this expression?
  - Two choices...

- (not True) or False
  - False or False
  - False

- not (True or False)
  - not True
  - False

Part 2

- ord("?") == 126 and ord("?") != 126
    - What does this mean?

- ord("?") evaluates to something
    - Numeric representation of "?"

- Either ord("?") == 126 or ord("?") != 126
    - Can't have it both ways

- Two possibilities
    - True and False
    - False and True

Part 3

```
x = 4
y = 10
if x > y:
    x = x + 1
    y = x * y
elif x < y:
    x = x - 1
    y = x * 2
else:
    x = y
    y = x
```

Part 3

```
x = 4
y = 10
if x > y:
    x = x + 1
    y = x * y
elif x < y:
    x = x - 1
    y = x * 2
else:
    x = y
    y = x
```

x = x - 1 → 4 - 1 → 3

y = x * 3 → 3 * 2 → 6

Part 4

```
def foo(u, v):
    sum = u + v
    prod = u * v
    ans = bar(sum, prod)
    return ans

def bar(x, y):
    z = 10 * x
    return y + z

def baz(n):
    m = foo(n + 1, n - 1):
    return n + m

a = baz(3)
```

Part 4

- What am I looking for in a stack diagram?

- Get the right answer
  - At least somewhere in the neighborhood

- Define all variables
  - Functions
  - Parameters

- Stack frames
  - Labeled
  - In order

Part 5

```
def mystery(x, y):
    """What do I do?"""

    difference = x - y
    if difference > 0:
        return x
    else:
        return y
```

Part 6

```
def stringChecker(string, element):
    """What do I do?"""

    if string == "":
        return False
    elif string[0] == element:
        return True
    else:
        return stringChecker(string[0:], element)
```

Part 7

- 0 is even
- 1 is odd
- An integer is even if the integer two numbers before is even

```
def isEven(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    else:
        return isEven(x-2)
```

Part 7

```
def isEven(x):
    if x == 0:
        return True
    elif x == 1:
        return False
    else:
        return isEven(x-2)
```

- Why do we need two base cases?
- Could we make do with just one?
    - Kind of...

Part 7

```python
def isEven(x):
    if x == 0:
        return True
    else:
        return isOdd(x-1)

def isOdd(x):
    if x == 0:
        return False
    else:
        return isEven(x-1)
```

Part 7

```
def isEven(x):
    if x / 2 == x / 2.0:
        return True
    else:
        return False
```

Cheeky...

Part 7

These functions are really inefficient

If you ever actually need to tell whether a number is even, just use the % operator

```
def isEven(x):
    return x % 2 == 0
```