The CS 5 Black Post

Penguins Invade Zoom Call

Claremont (AP)—A wild group of penguins took over a Zoom-based lab session yesterday, sharing screenshots of icebergs and dead fish. Taking advantage of Zoom’s “remote control” feature, they managed to interfere with students who were diligently attempting to complete their CS 5 homework. “I had just gotten my program to work,” sobbed one student, they activated my ‘delete’ key and wiped it all out! Now I have to start all over. It took me hours to get my program to say ‘Hello, world’

Read Section 2.8 (easy peasy!)

Booleans

```python
def dbl(x):
    return 2 * x
def trbl(x):
    return 2 * x

def happy(yay):
    y = dbl(yay)
    return y + 42
def sad(boo):
    y = trbl(boo)
    return y + 42
def friendly(pal):
    y = dbl(pal)
    print(y, "is very nice!")
    return y + 42
```

The “Truth” about Python’s Booleans

```python
>>> 3 == 1+2
True
>>> 42 == "ham"
False
>>> "spam" > "ham"
True
>>> 42 > "spam"
Barf!
```

return vs print...

```
George Boole
1815-1864
```

```python
>>> True + 41
42
>>> 2**False == True
True
```

Strings are in single or double quotes

Demonstrating the True “power” of Falsity!
Lists!

>>> L = [1, 42, 3, 4]
>>> L
[1, 42, 3, 4]
>>> L + 10
Traceback (most recent call last):
  File "<stdin>" line 1 in ?
TypeError: can only concatenate list (not "int") to list

>>> L + [50]
[1, 42, 3, 4, 50]
>>> L
[1, 42, 3, 4]  # L doesn't change!

>>> L*2
[1, 42, 3, 4, 1, 42, 3, 4]

>>> M = [42, "hello", 3+2j, 3.141, [1, 2, 3, 4, 5, 6]]

List Indexing and Slicing!

>>> M = [42, 3, 98, 37]
>>> M[0]
42
>>> M[2]
98
>>> M[0:2]
[42, 3]
>>> M[0:3:2]
[42, 3]
>>> M[1:]
[3, 98, 37]
>>> M[:-1]
[42, 3, 98]
>>> M[1:-2]
[3, 98]

Python slices just like slapchop!

0 1 2 3

What kind of thing does this return?

Try to reverse the list!

Strings Revisited

>>> S = "I love Spam!"

>>> S[0]
'I'

>>> S[11]
'

>>> S[2:6]
'

>>> S[11:6:-1]
'

>>> S*2

Hey penguins, get off my slides!

if, else...

def special(x):
    """This function demonstrates the use of if and else""
    if x == 42:
        return "Very special number!"
    else:
        return "Stupid, boring number."

def special(x):
    if x == 42:
        return "Very special number!"
    return "Stupid, boring number."

Altematively??

Notice how lines with the same level of indentation are in the same code block!
if, elif, else...

```python
def superSpecial(x):
    """This function demonstrates the use of if, elif, and else""
    if x < 41:
        ans = "Small number"
    elif x == 42 or x % 42 == 0:
        ans = "Nice!"
    elif 41 <= x <= 43:
        ans = "So close!"
    else:
        # We might do more stuff here...
        ans = "Yuck!"
    return ans
```

Would swapping the order of these elifs give the same behavior?

Avoiding elif

```python
def unwise(x):
    """This function avoids using elif""
    if x == 42 or x % 42 == 0:
        ans = "Nice!"
    if 41 <= x <= 43 or x != 42:
        ans = "So close!"
    if x != 42 or x % 42 != 0:
        ans = "Yuck!"
    return ans
```

Is Recursion Magic?

```python
factorial(3):
    return 3 * factorial(2)
    return 2 * factorial(1)
    return 1 * factorial(0)
```

"To understand recursion, you must first understand recursion"—anonymous Mudd alum

A Tower of Fun!

Math

```python
tower(3) = 2^2^2
tower(4) = 2^{2^{2^2}}
tower(5) =
```

Python (Functional)

```python
# recursive factorial
def factorial(n):
    '''This computes n!'"
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

The tower function is taking recursion to new heights!
Computing the Length of a List

```python
>>> len([1, 42, "spam"败])
3
>>> len([1, [2, [3, 4]]])
```

```python
def len(List):
    '''Returns the length of List'''
```

Reversing a List

```python
>>> reverse([1, 2, 3, 4])
[4, 3, 2, 1]
```

```python
def reverse(L):
    '''Returns a new list that is the reverse of the input list'''
```

Reversing a List

```python
>>> reverse([1, [2, [4, 5], 6], 7])
```

Deep-Reversing a List

```python
>>> reverse([1, [2, [4, 5], 6], 7])
[7, [2, [4, 5], 6], 1]
```

```python
>>> deepReverse([1, [2, [4, 5], 6], 7])
[7, [6, [5, 4], 2], 1])
```

This definitely requires recursion! Fun problem on this week’s HW!
Recursion = :^)

Recursion, conditional statements, and lists suffice to give us a Turing-complete programming language!

Variables, assignment (=), for, while, etc. are all unnecessary!