Penguin Leads Surfing Trip

Oceanside (UPI)—In an attempt to make up for the chaos caused by her ill-behaved colleagues during a Harvey Mudd College, a well-tanned penguin took a group of first-year students on a beach outing in which they learned to surf. “It was awesome,” gushed one surfer. “I stood up the very first time and I would have made it all the way in except that I slipped on some fish oil.”

Another beginner lauded the experience even though he returned with gashes on his chin. “I got to see the ocean from underneath!” he exclaimed. “I had no idea there were fish under there!”

def even(x):
    '''Returns True iff x is even'''
    return x % 2 == 0

>>> list(filter(even, range(100)))
[0, 2, 4, 6, …, 98]

def short(List):
    '''Returns True iff List has len <= 2'''
    return len(List) <= 2

>>> list(filter(short, [["spam", "yum"], [42], [1, 2, 3]]))
Functions are Data

def divides(n):
    def div(k):
        return n % k == 0
    return div

>>> div10 = divides(10)
>>> div10(2)

>>> listOfFunctions = [divides(10), divides(20)]
>>> listOfFunctions[0](2)

Anonymous Functions

filter(lambda x: x%2 == 0, range(100))

>>> lambda_dbl = lambda x: 2 * x
>>> lambda_dbl(21)
42

Lambda aka “anonymous functions”

>>> list(filter(lambda x: x%2 == 0, range(100)))

>>> list(filter(lambda List: len(List) <= 2, [[["spam", "yum"], [42], [1, 2, 3]]]))

Lambda

even = lambda x: x%2 == 0
def even(x):
    '''Returns True iff x is even'''
    return x % 2 == 0

short = lambda List: len(List) <= 2
def short(List):
    '''Returns True iff List has len <= 2'''
    return len(List) <= 2
def ugly(item, L):
    newL = list(map(lambda x: x == item, L))
    return sum(newL) > 0

This is exploiting the fact that True == 1 and False == 0.

from functools import reduce
def mystery(item, L):
    newL = list(map(lambda x: x == item, L))
    return reduce(lambda x, y: x or y, newL)

MUCH better!

A Prime Example

Write a function called prime(n) that returns True if n is prime and False otherwise by testing all possible divisors from 2 to n-1 (or sqrt of n)

def prime(n):
    possibleDivisors = range(2, n)
    divisors = filter(lambda X: n % X == 0, possibleDivisors)
    return ???

A version of this was an extra-credit problem in Homework 0!

The Alien’s Life Advice

Ask lots of questions!

That’s how Hermione learned so much!
Use-It-Or-Lose-It

Power Set!

>>> powerset([1, 2])
[[], [2], [1], [1, 2]]

>>> powerset([1, 2, 3])
[[], [3], [2], [2, 3], [1], [1, 3], [1, 2], [1, 2, 3]]

>>> powerset([1])

>>> powerset([])

This really demonstrates the power of functional programming!
The order in which the subsets are presented is unimportant but within each subset, the order should be consistent with the input set.

Power Set!
def powerset(L):

The Knapsack Problem...

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>Tofu</td>
<td>3</td>
<td>112</td>
</tr>
<tr>
<td>Chocolate</td>
<td>4</td>
<td>125</td>
</tr>
</tbody>
</table>

Kingdom of Shmorbodia

Knapsack Capacity: 5? 6? 7?

>>> knapsack(7, [[2, 100], [3, 112], [4, 125]])
237

Prof. I. Lai thinks that a "greedy solution" is the way to go!

Worksheet and Demo
The Knapsack Revisited…

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>Tofu</td>
<td>3</td>
<td>112</td>
</tr>
<tr>
<td>Chocolate</td>
<td>4</td>
<td>125</td>
</tr>
</tbody>
</table>

Knapsack Capacity: 5? 6? 7?

```python
>>> knapsack(7, [[2, 100], [3, 112], [4, 125]])
[[2, 100], [3, 112], [4, 125]]
```

Comparing DNA via Longest Common Subsequence (LCS)

```
>>> LCS("AGGACAT", "ATTACGAT")
5
```

Try this in your notes!

Recursive Approach…

```python
def LCS(S1, S2):
    if BASE CASE
        return
    else:
        LCS("spam", "sam!")
```

Solution follows