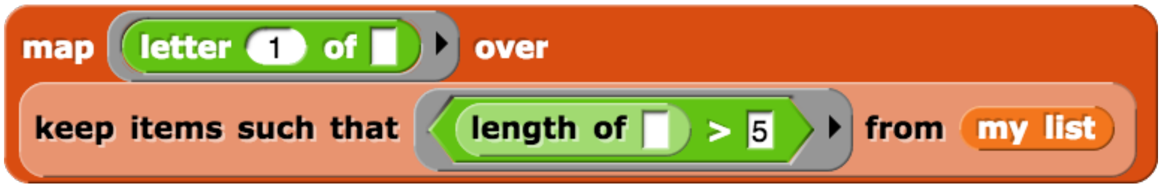
Discussion 13: Python III

Warm Up: Starting List Comprehensions

1. Translate the following Snap! code to a Python list comprehension.



List Comprehension Practice

```
fav_numbers = {'Schuyler': 120, 'Matthew': 12, 'Mansi': 7}
nums = [10, 12]
```

1. Use a list comprehension to return a list of the TAs whose favorite numbers are in `nums`.

---

2. Write a list comprehension over `list_of_lists`, which is a list of lists, where each sublist is composed of numbers, that finds the sum of each sublist. To find the sum of a list, you can call `sum` on the list.

```
>>> sum([1, 2])
3
```

List Comprehension Practice

In the following questions, we will be writing code to find the index of an item in a list, both recursively and iteratively. For the purposes of this question, you may assume that the item you are looking for appears in the list exactly once.

1. Write the function described above recursively in Python.

```
>>> find_index(5, [2, 5, 7])
1

def find_index(item, lst):
```

2. Now, write it using a list comprehension. You may refer to the item you are looking for with the variable `item`. It may be easier to first write out a for loop to accomplish this task, and then write it as a list comprehension.

____

Lambda Functions

1. Write a lambda function called f that takes in a number and outputs that number squared.
f = ____

2. Now, use a list comprehension and your lambda function f to return a list the squares of all numbers between 1-5, inclusive.

____

Challenge

What would the interpreter display for the following lines of code?

```
>>> S = “Berkeley”
>>> S[1:3]
```

____

```
>>> [x * 2 for x in range(4) if x % 2 == 1]
```

____

```
>>> "".join([word[0] for word in "Univ of Calif at Berkeley".split() if
not(len(word) == 2)])
```

____

```
>>> "".join([word[0] for word in "Univ of Calif at Berkeley" if not(len(word) ==
2)])
```

____

```
>>> f1 = lambda x: x + x
>>> f2 = lambda x: x > 9
>>> [f(10) for f in [f1, f2]]
```

____

```
>>> y = 3
>>> f = lambda x: lambda: x + y
>>> f(2)()
```

____

```
>>> g = lambda y: x + y
>>> g(2)
```

____