

Discussion 12: Python II

Planning Your Phase I

1. In the table below, write out Python code to execute the following commands on `my_dict`.

```
my_dict = {'Math': '1A', 'English': 'R1A'}
```

Add the key 'CS' with the value '10'	<code>my_dict['CS'] = '10'</code>
Access the value of 'Math'	<code>my_dict['Math']</code>
Change the value of 'Math' to '1B'	<code>my_dict['Math'] = '1B'</code>
Check if 'UGBA' is a key in <code>my_dict</code>	<code>'UGBA' in my_dict</code> (can also say <code>'UGBA' in my_dict.keys()</code>)
Check if '10' is a value in <code>my_dict</code>	<code>'10' in my_dict.values()</code>
Get a list of the keys in <code>my_dict</code>	<code>list(my_dict.keys())</code>

2. Can you access a key, value pair in a dictionary by its index? [No, dictionaries do not have indices.](#)

3. Are keys or values in a dictionary returned in a predictable order? [No.](#)

4. Can dictionaries have duplicate keys? [No.](#)

Dictionary Practice

```
fav_numbers = {'Schuyler': 120, 'Matthew': 12, 'Mansi': 7}
```

1. Increment each person's favorite number by the length of their name.

```
for person in fav_numbers:  
    fav_numbers[person] += len(person)
```

2. Print a list of all people from `fav_numbers` whose favorite numbers are even.

```
names = []  
for person in fav_numbers:  
    if fav_numbers[person] % 2 == 0:  
        names.append(person)  
print(names)
```

3. Write a function that merges two dictionaries. Both dictionaries will have strings as keys and numbers as values. If you come across a key that's present in both dictionaries, add together its values in the merged dictionary.

```
>>> dict1 = {'Schuyler': 10, 'Matthew': 15}
>>> dict2 = {'Matthew': 5, 'Mansi': 4}
>>> merge_dicts(dict1, dict2)
{'Schuyler': 10, 'Matthew': 20, 'Mansi': 4}
```

```
def merge_dicts(d1, d2):
    new_dict = {}
    for key in d1:
        new_dict[key] = d1[key]
    for key in d2:
        if key in new_dict:
            new_dict[key] += d2[key]
        else:
            new_dict[key] = d2[key]
    return new_dict
```

4. Assume we have executed the code below in the Python interpreter. What will be displayed after each of the following code snippets executes? If the result is an error message, just write "Error." Assume that these lines are executed independently, NOT sequentially. (Continued on the next page)

```
>>> food_dict = {"fruit": "apple", "veggie": "carrot", "beverage": "water",
"grain": "rice"}
>>> len(food_dict)
4
>>> list(food_dict)
['beverage', 'fruit', 'grain', 'veggie']
# you are not expected to return them in the correct order

>>> food_dict[0]
Error

>>> ('fruit' in food_dict) and ('apple' in food_dict)
False

>>> ("fruit" in food_dict.keys()) and ("apple" in food_dict.values())
True

>>> for food in food_dict:
...     food += "s"
```

```
>>> food_dict
{'beverage': 'water', 'fruit': 'apple', 'grain': 'rice', 'veggie': 'carrot'}
# you are not expected to get the correct order
```

Explanation:

You are only adding 's' to the food variable, not the actual key in the
dictionary. Strings, which the keys are, are not mutable.

```
>>> def recursion_is_fun(dict1, dict2):
...     if dict2 == {}:
...         return dict1
...     dict2.pop(list(dict2)[0])
...     return recursion_is_fun(dict1, dict2)
```

```
>>> copy = food_dict
>>> recursion_is_fun(food_dict, copy)
{}
```

Explanation:

To make a copy of a dictionary in Python, you must say dict_name.copy()

```
>>> more_food = {"protein" : "chicken"}
>>> food_dict["more food"] = more_food
>>> food_dict
```

```
{'beverage': 'water', 'more food': {'protein': 'chicken'}, 'fruit': 'apple',
'grain': 'rice', 'veggie': 'carrot'}
# you are not expected to get the correct order
```

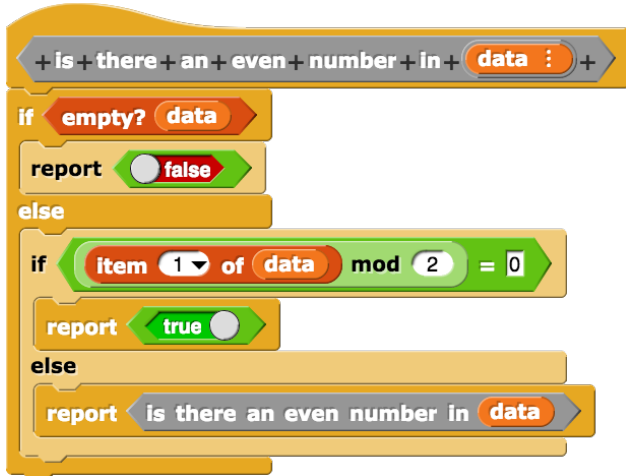
Recursion in Python

1. In the table below, translate the Snap! code to Python. This Python code will be useful in writing recursive functions in Python.

all but first of my list	my_list[1:]
all but last of my list Note: This block doesn't actually exist in Snap!	my_list[:-1]
all but first of all but last of my list	my_list[1:-1]
5 in front of my list	[5] + my_list
append my list your list	my_list + your_list

Note: The Python syntax for the first three rows would be the same whether your variable is a list or a string.

2. Translate the following code from Snap! to Python:



```
is_even_in(lst):  
    if len(lst) == 0:  
        return False  
    elif lst[0] % 2 == 0:  
        return True  
    return is_even_in(lst[1:])
```

3. Write a recursive function in Python that removes certain items from a list, as described below. It should create a new list, not mutate the input list.

```
>>> delete_elements(4, [4, 5, 6])  
[5, 6]  
>>> delete_elements(7, [4, 5, 6])  
[4, 5, 6]  
>>> delete_elements(4, [4, 5, 6, 4])  
[5, 6]
```

```
def delete_elements(x, lst):  
    if len(lst) == 0:  
        return lst  
    if lst[0] == x:  
        return delete_elements(x, lst[1:])  
    else:  
        return [lst[0]] + delete_elements(x, lst[1:])
```