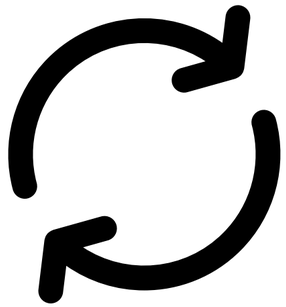
THE UNIVERSITY OF MELBOURNE

COMP10001 - Sem 2 2024 - Week 5

Foundations of Computing

Daksh Agrawal

Loops



```
for i in [5,10,20]:
    print(i)
```

range(0, 10, 2):

```
for number in range(0,10):
    print(number)
```

```
x = 4
while x > 0:
    print(x)
    x = x - 1
print(x)
```

1

4, 3, 2, 1

0

```
i = 2
while i < 8:
    print(f"The square of {i} is {i * i}")
    i = i + 2
```

The square of 2 is 4
4 is 16
6 is 36

```
for ingredient in ("corn", "pear", "chilli", "fish"):
    if ingredient.startswith('c'):
        print(ingredient, "is delicious!")
    else:
        print(ingredient, "is not!")
```

\n ⇒ newline

```
i = 0
colours = ("pink", "red", "blue", "gold", "red")
while i < len(colours):
    if colours[i] == "red":
        print("Found red at index", i)
    i += 1
```

Found red at index 1
index 4

\n ⇒ newline

4+3+2
3+1+1
0+4+5+7

```
input_str = "4+3+2\n3+1+1\n0+4+5+7"
for line in input_str.split("\n"):
    num_sum = 0
    for num in line.split("+"):
        num_sum += int(num)
    print(num_sum)
```

["4", "3", "2"]

9
5
16

```
count = 0
items = ('eggs', 'spam', 'more eggs')
while count < len(items):
    print(f"need to buy {items[count]}")
    count += 1
```

```
items = ('eggs', 'spam', 'more eggs')
for count in range(len(items)):
    print(f"need to buy {items[count]}")
```

```
items = ('eggs', 'spam', 'more eggs')
for item in items:
    print(f"need to buy {item}")
```

Rewrite using for loops

```
i = 2
while i < 8:
    print(f"The square of {i} is {i * i}")
    i = i + 2
```

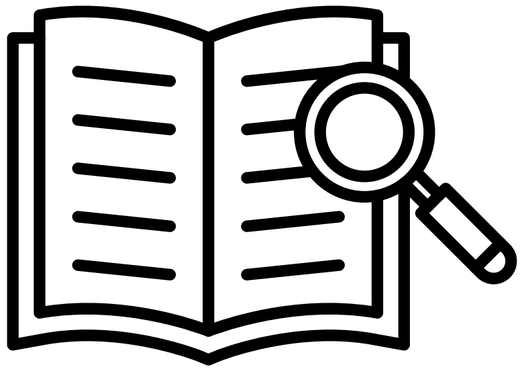
```
for i in range(2,8,2):
    print(...)
```

Rewrite using while loops

```
for ingredient in ("corn", "pear", "chilli", "fish"):
    if ingredient.startswith('c'):
        print(ingredient, "is delicious!")
    else:
        print(ingredient, "is not!")
```

```
i = 0
ingredients = (...)
while i < len(ingredients):
    ingredient = ingredients[i]
    if ingredient.startswith("c"):
        print(ingredient "is delicious")
    else:
    i += 1
```

Dictionaries



```
subjects = {
    ”COMP10001": ”Foundations of Computing”,
    “COMP10002”: ”Foundations of Algorithms”,
    “MAST10006”: “Calculus 2”,
    “MAST10007”: “Linear Algebra”
}
```

```
subjects = {
    "COMP10001": "Foundations of Computing",
    "COMP10002": "Foundations of Algorithms",
    "MAST10006": "Calculus 2",
    "MAST10007": "Linear Algebra"
}

subjects["COMP10001"]
subjects["COMP10001"] = "Intro. to Programming"
```

"Intro to. Programming"

Foundations of Computing

```
subjects = {
    ”COMP10001": ”Foundations of Computing”,
    “COMP10002”: ”Foundations of Algorithms”,
    “MAST10006”: “Calculus 2”,
    “MAST10007”: “Linear Algebra”
}

subjects.keys()
[”COMP10001", “COMP10002”, “MAST10006”, “MAST10007”]

subjects.values()
[”Foundations of Computing”, ”Foundations of Algorithms”, “Calculus 2”, “Linear
Algebra”]
```

```
subjects = {
    ”COMP10001": ”Foundations of Computing”,
    “COMP10002”: ”Foundations of Algorithms”,
    “MAST10006”: “Calculus 2”,
    “MAST10007”: “Linear Algebra”
}
```

list →

key

value

tuple

```
subjects.items()
[
 (”COMP10001", ”Foundations of Computing”),
 (“COMP10002”, ”Foundations of Algorithms”),
 (“MAST10006”, “Calculus 2”),
 (“MAST10007”, “Linear Algebra”)
]
```

Evaluate by Hand, given

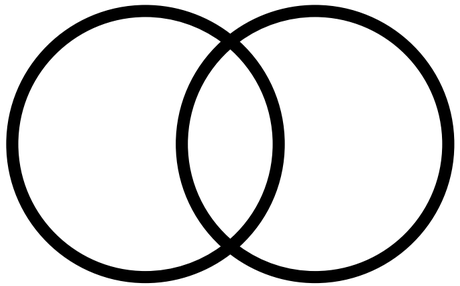
```
d = {"R": 0, "G": 255, "B": 0, "other": {"opacity": 0.6}, "A": 50}
```

255

{"opacity": 0.6, "blur": 0.13

`"R" in d` "R" in d.keys() True	`d["R"]` 0	`d["R"] = 255`	`d["A"]` Error KeyError)
`d["A"] = 50`	`x = d.pop("G")` 255 int	`d["other"]["blur"] = 0.1`	`d.items()` [("R", 255), ("B", 0), ("other", {"opacity": 0.6, "blur": 0.13}), ("A", 50)]

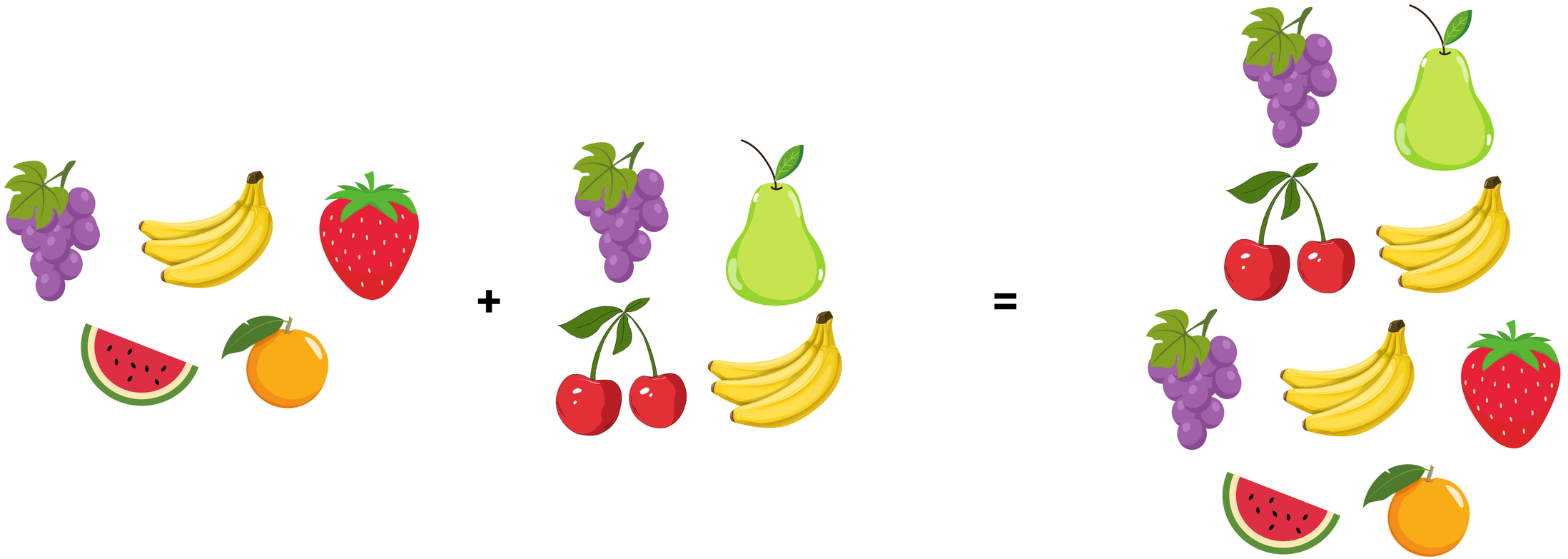
Sets



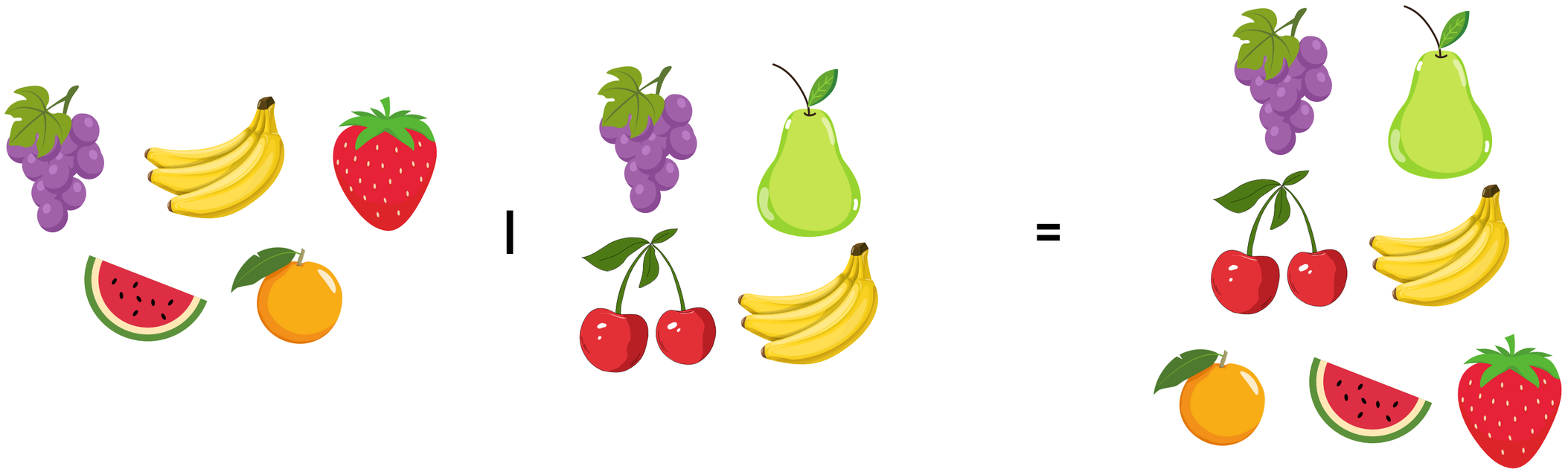
{4, 7, 8, 10}

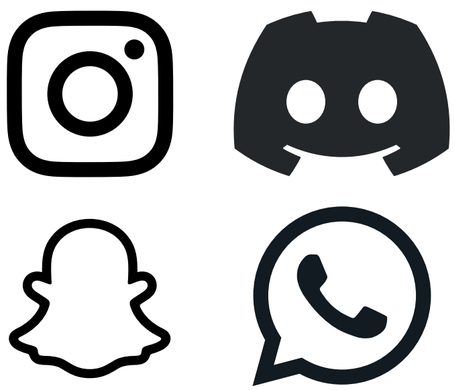
{4, 7, 8, 10, 4} 

Simple Lists



Set Union





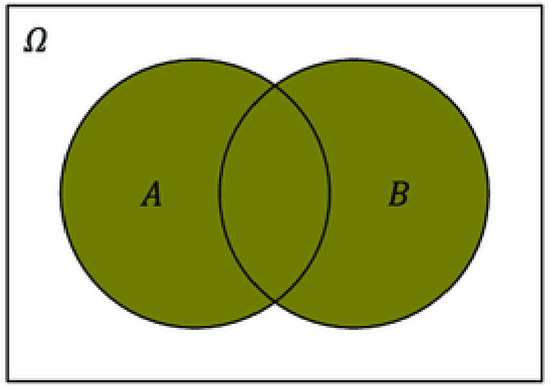
&

&

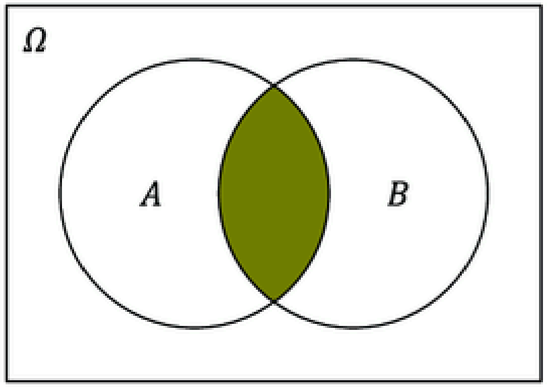
=

Set Intersection

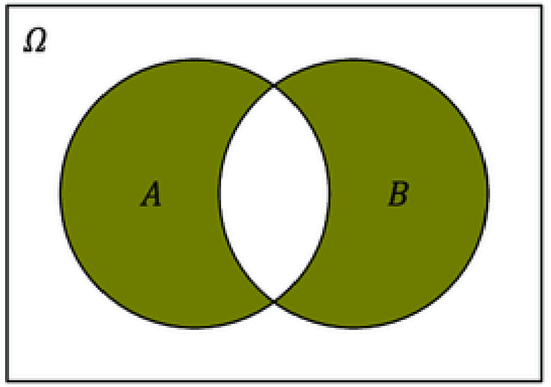
$A \cup B$



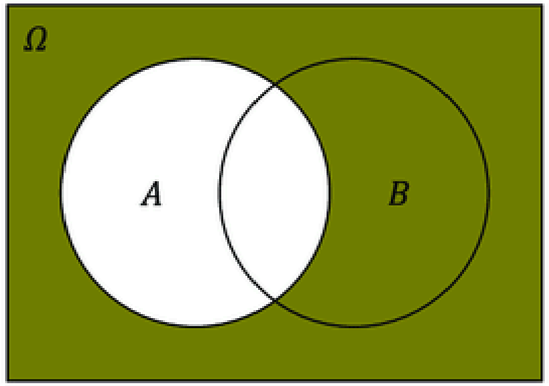
$A \cap B$



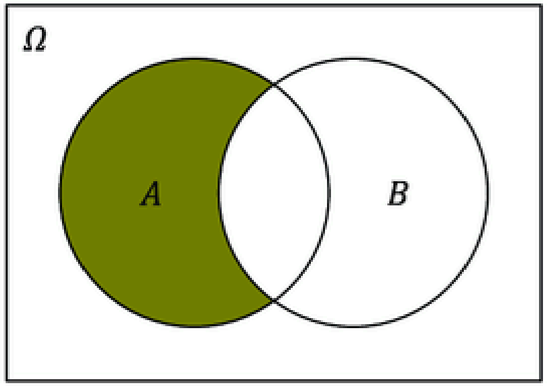
$A \Delta B = (A \cup B) - (A \cap B)$



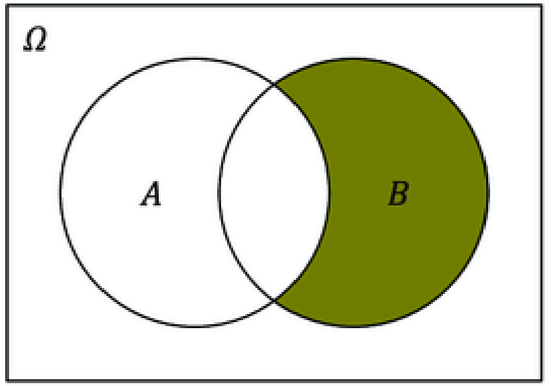
A'



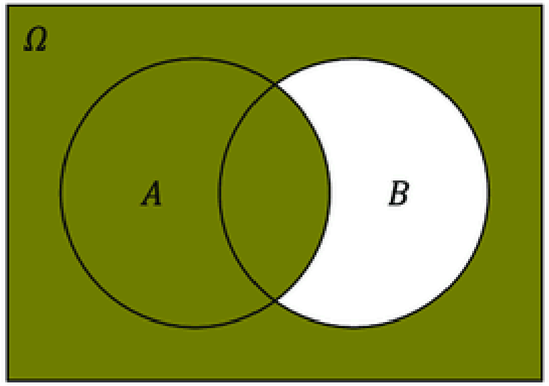
$A \cap B'$



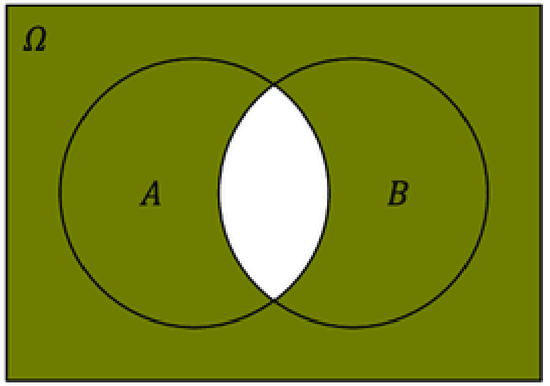
$A' \cap B$



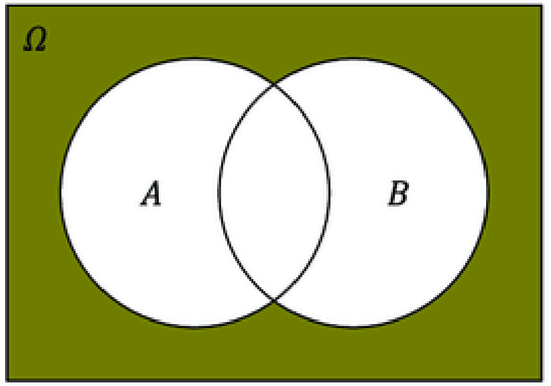
$A \cup B'$



$A' \cup B' = (A \cap B)'$



$A' \cap B' = (A \cup B)'$



Evaluate, given s1 = {1, 2, 4} and s2 = {3, 4, 5}

`s1.add(7)` S1 = {1, 2, 4, 7}	`s1.add(2)` S1 = {1, 2, 4}	`s2.remove(5)` S2 = {3, 4}

`s1 & s2`
`s1.intersection(s2)`
{4}
	`s1 | s2`
	`s1.union(s2)`
	{1, 2, 4, 3, 5}
		`s1 - s2`
		{2, 1}

s1 = {1, 2, 3}

print(s1) = {3, 1, 2}

Sets and Dictionaries have NO order

Don't sort

Paper Programming

Write a function which takes a tuple of strings and returns a list containing only the strings which contain at least one exclamation mark or asterisk symbol. **words_with_symbols(('hi', 'there!', '*_*'))** should return ['there!', '*_*'].

```
def words_with_symbols(words):
    answer = []
    for word in words:
        if "!" in word or "*" in word:
            answer.append(word)
    return answer
```

Write a function **sort_by_score(player_scores)** that takes a dictionary containing a player name as the key and their score as the value, and returns a list of **(score, player_name)** tuples sorted by highest to lowest score.

```
player_scores = {'Sonic': 299, 'Zelda': 421, 'Mario': 367, 'Pikachu': 152}
print(sort_by_score(player_scores))
```

Should output:

```
[(421, 'Zelda'), (367, 'Mario'), (299, 'Sonic'), (152, 'Pikachu')]
```

```
def sort_by_score(player_scores):
    score_items = player_scores.items()
    for i in range(len(score_items)):
        score_items[i] = score_items[i].reversed()
    score_items.sort(reversed=True)
    return score_items
```

```
list.reverse()
r_list = list.reversed()
sort()
sorted()
```

Write a function which takes a string as input and prints the frequency of each character in the string using a dictionary. **freq_counts('booboo')** should print:

b 2
o 4

```
def freq_counts(text):
    freq = {}
    for i in text:
        if i in freq:
            freq[i] = freq[i] + 1
        else:
            freq[i] = 1
```

{'b': 2, 'o': 4}

```
for key in freq.keys():
    print(key, freq[key])
```

Write a function which takes a string, a character and an integer threshold and returns **True** if the character appears in the string with a frequency above the threshold, **False** if it appears at or below the threshold, and **None** if it doesn't appear at all. **above_thresh('I like the letter e', 'e', 3)** should return **True**.

```
def above_thresh(text, letter, t):
    freq = text.count(letter)
    if freq > t:
        return True
    elif freq == 0:
        return None
    else:
        return False
```

Write a function called **decode(key1, key2, ciphertext)** that takes two string keys and a string **ciphertext** to decode. To decode it:

- **Even indices of ciphertext:** If the character at this index of the **ciphertext** is in **key1** but not in **key2** then skip it, otherwise add it to the cleartext.
- **Odd indices of ciphertext:** If the character at this index of the **ciphertext** is in **key2** but not in **key1** then skip it, otherwise add it to the cleartext.

Your function should return the cleartext (decoded) string at the end.

```
key1 = "a01g4ds4?5atpv.qy52"
key2 = "asb8gh.dvt7xyz1mz3"
ciphertext = "0y5mpzpxpoquq 0s4zqhoh5l5hqv?eqh2xp8qx03p85hd3?m0x?zqz5b mim2bt!"
print(decode(key1, key2, ciphertext))
```

The above code should print out a readable message!

Shape of You (Ascii Remix)

Give each other a shape, and try to draw it using ASCII.

Anonymus Feedback