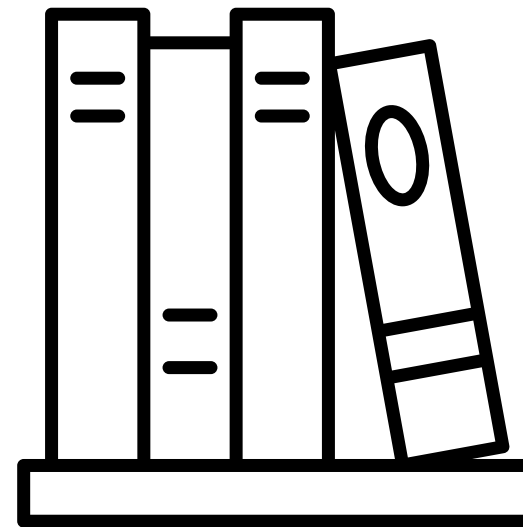


COMP10001 - Sem 2 2024 - Week 8

Foundations of Computing



Daksh Agrawal



Libraries

Default Dictionaries

```
1 text = "This is a test 123 456 789"
2
3 freq = {}
4 for char in text:
5     if char in freq:
6         freq[char] += 1
7     else:
8         freq[char] = 1
```

```
1 from collections import defaultdict
2
3 text = "This is a test 123 456 789"
4
5 freq = defaultdict(int)
6 for word in text.split():
7     freq[word] += 1
```

Rewrite the following with a default dictionary

```
my_dict = {}  
for i in range(10):  
    if i % 3 in my_dict:  
        my_dict[i % 3].append(i)  
    else:  
        my_dict[i % 3] = [i]
```

.

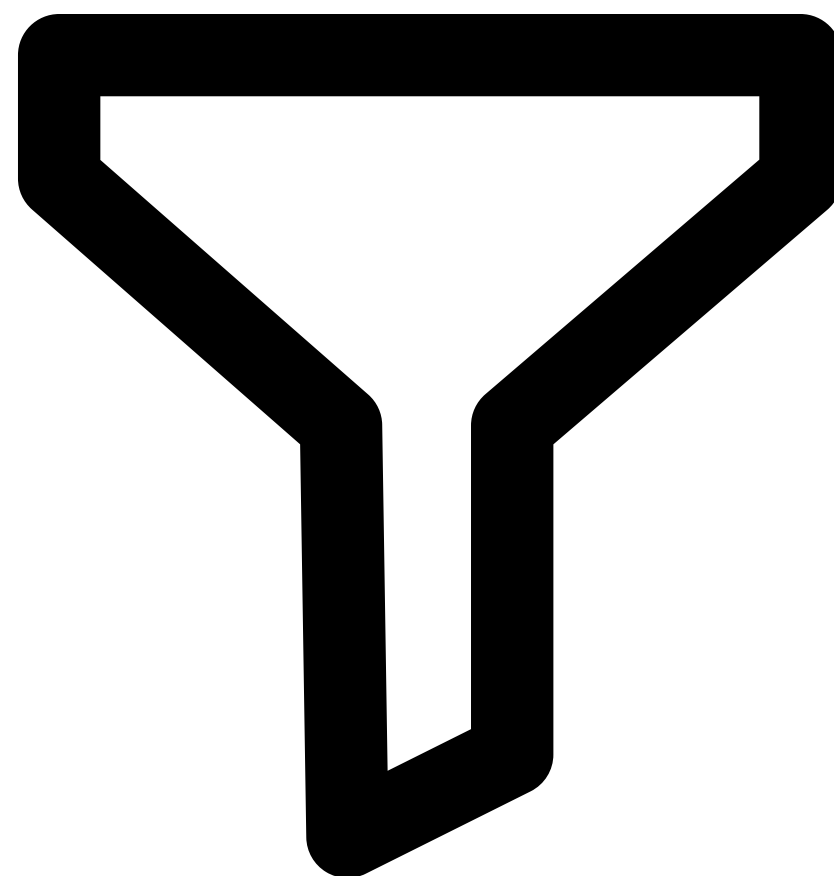
Assign the value of the square root of 2 to var using three different methods, with each method using one of the following ways of import.

```
import math
```

```
from math import sqrt
```

```
from math import sqrt as square_root
```

λ



```
1 def square(x):  
2     return x * x  
3  
4  
5 square_lambda = lambda x: x * x
```

```
1  nums = [1, 2, 3, 4, 5]
2  square_lambda = lambda x: x * x
3  squares = map(square_lambda, nums)
4  squares = map(lambda x: x * x, nums)
```



```
1      nums = [1, 2, 3, 4, 5]
2
3
4      1 usage  new *
5      def square(x):
6          return x * x
7
8      squares = []
9      for i in nums:
10         squares.append(square(i))
```

```
1      nums = [1, 2, 3, 4, 5]
2
3      squares = map(lambda x: x ** 2, nums)
```

```
1  nums = [1, 2, 3, 4, 5]
2
3
4  1 usage  new *
5  def is_even(num):
6      return num % 2 == 0
7
8  even_nums = []
9
10 for num in nums:
11     if is_even(num):
12         even_nums.append(num)
```

```
1  nums = [1, 2, 3, 4, 5]
2
3  even_nums = filter(lambda x: x % 2 == 0, nums)
```

```
food_list = ["sushi", "pizza", "hot pot", "fish and chips", "burgers"]
```

```
sorted(food_list, key=lambda x:x[-1])
```


```
list(filter(lambda x: len(x.split()) == 1, food_list))
```

```
def price(food):  
    return len(food) * 2  
list(map(price, food_list))
```



Errors

5 × × 6



SyntaxError: invalid syntax

`NameError: name 'a' is not defined`

`TypeError: must be str, not int`

6/0

`ZeroDivisionError: division by zero`

`IndexError: list index out of range`

$$x^2 = (y + z)^2$$

$$x^2 = y^2 + z^2$$

```
1  def disemvowel(text):
2      """ Returns string `text` with all vowels removed """
3      vowels = ('a', 'e', 'i', 'o', 'u')
4      answer = text[0]
5      for char in text:
6          if char.lower() is not in vowels:
7              answer = char + answer
8      print(answer)
```



```
11 def big-ratio(nums, n):  
12     """ Calculates and returns the ratio of numbers  
13 in non-empty list `nums` which are larger than `n` """  
14     n = 0  
15     greater_n = 0  
16     for number in nums:  
17         if number > n:  
18             greater_n += 1  
19             total += 1  
20     return greater_n / total  
21  
22     nums = [4, 5, 6]  
23     low = 4  
24     print(f"{100*big_ratio(nums, low)}% of numbers are greater than {low}")
```

1. In this task we will play with numbers again. For example, the number 89 has two digits - 8 at the first position and 9 at the second position. If we raise each digit of 89 to the power of its position and take a sum: $8^1 + 9^2$, the answer is 89 itself! We call the number with this property a *cool number*. Another example of a cool number would be 598 because $5^1 + 9^2 + 8^3 = 598$.

Write a function `get_cool_number(n)` that takes a positive integer n and returns the n^{th} cool number, starting from 1. That is, the first nine cool numbers are 1 to 9, then 10 is not cool, 11 is not cool ... we don't see another cool number (the 10th one) until 89, then the 11th cool number is 135, and so on. Therefore, your function `get_cool_number(10)` should return 89.



ONEREPUBLIC

NATIVE

COUNTING STARS

Data Preprocessing

```
26 reviews = [  
27     ("5", "Absolutely love the energy of this song! It's a classic."),  
28     ✖ ("4", "Great melody and lyrics, but gets a bit repetitive after a while."),  
29     ("3", "Good song, but not my favorite. The chorus is catchy though."),  
30     ("5", "An all-time favorite! Can't stop humming along."),  
31     ("2", "Didn't really connect with it, but the beat is okay."),  
32     ("4", "Solid track with a nice rhythm. Perfect for road trips."),  
33     ("3", "It's decent, but I feel like it could have been more powerful."),  
34     ✖ ("5", "This song never gets old! The lyrics are so meaningful."),  
35     ("4", "Love the vibe! It's uplifting and fun to sing along to."),  
36     ("3", "It's a good song, but not something I would listen to on repeat."),  
37 ]
```