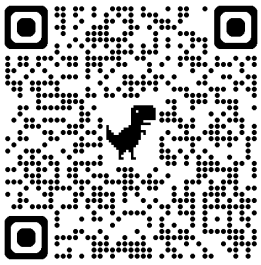
UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
Computer Science Division

CS10 Fall 2025 **TA: Victoria**



Discussion 12: Data Structures in Python + Concurrency

Instructions:

- If you're attending this section in-person, please log into iClicker!
- If you missed this discussion, fill out this entire worksheet, and upload it to the Gradescope assignment titled "Discussion 12" by next Discussion.
- For the worksheet, you can either explain the process in words, show a screenshot, or draw the block/process.
- For today's discussion, you are welcome to use the following starter file: **tinyurl.com/fa25-disc12-starter**
- Please complete the Feedback Form **tinyurl.com/fa25-disc-form**

Group Activity / Question of the Day

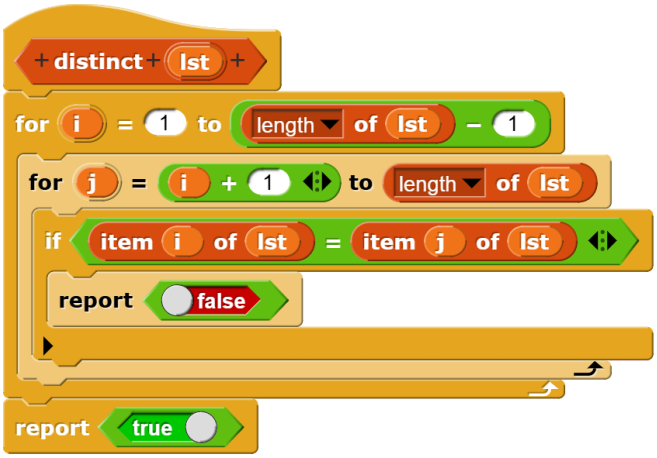
- Do you have any specific things you do before an exam (i.e. get a specific drink, watch a certain show, listen to a pump-up jam, etc.)

Required (Pages 2 - 5):

Section I - Translate from Snap! to Python

---

1. Translate the following Snap! code to its Python equivalent:



2. Translate the following Snap! blocks to Python using list comprehensions:

Hint: Combine using the _ + _ in Snap! can be done using the sum(lst) function in Python

Section II - Data Structures

---

1. Fill out the table below with the appropriate Python syntax:

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

Add the key 'CS' with the value '10'	
Access the value of 'Math'	
Change the value of 'Math' to '1B'	
Check if 'UGBA' is a key in class_dict	
Check if '10' is a value in class_dict	
Get a list of the keys in class_dict	

2. After executing the following code, what is the value of `names`?

```
gifts = [0, 1, 2, 5, 3, 2]
names = ["Yishu", "Alonzo", 3, 4, 2, "Vedansh", 5,
"President", "Mehul"]

for gift in gifts:
    if gift < len(names):
        names.pop(gift)
    else:
        names.append(gift)
```

3. Write a function called `merge_dicts` that takes in two dictionaries as inputs and returns a new dictionary as output that contains all entries from both input dictionaries. You can assume that both dictionaries have Strings as keys and Integers as values. For any keys present in both dictionaries, the corresponding value in the output dictionary should be the sum of the values from the inputs. You cannot assume both dictionaries are the same length.

4. Write a function called `remove_vowels` that takes in a tuple of strings as an input and returns a new tuple as an output that removes all vowels from each string in the output tuple. *Hint: Tuples are immutable, but you can always cast a data type like a list to a tuple.*

Section III - Concurrency

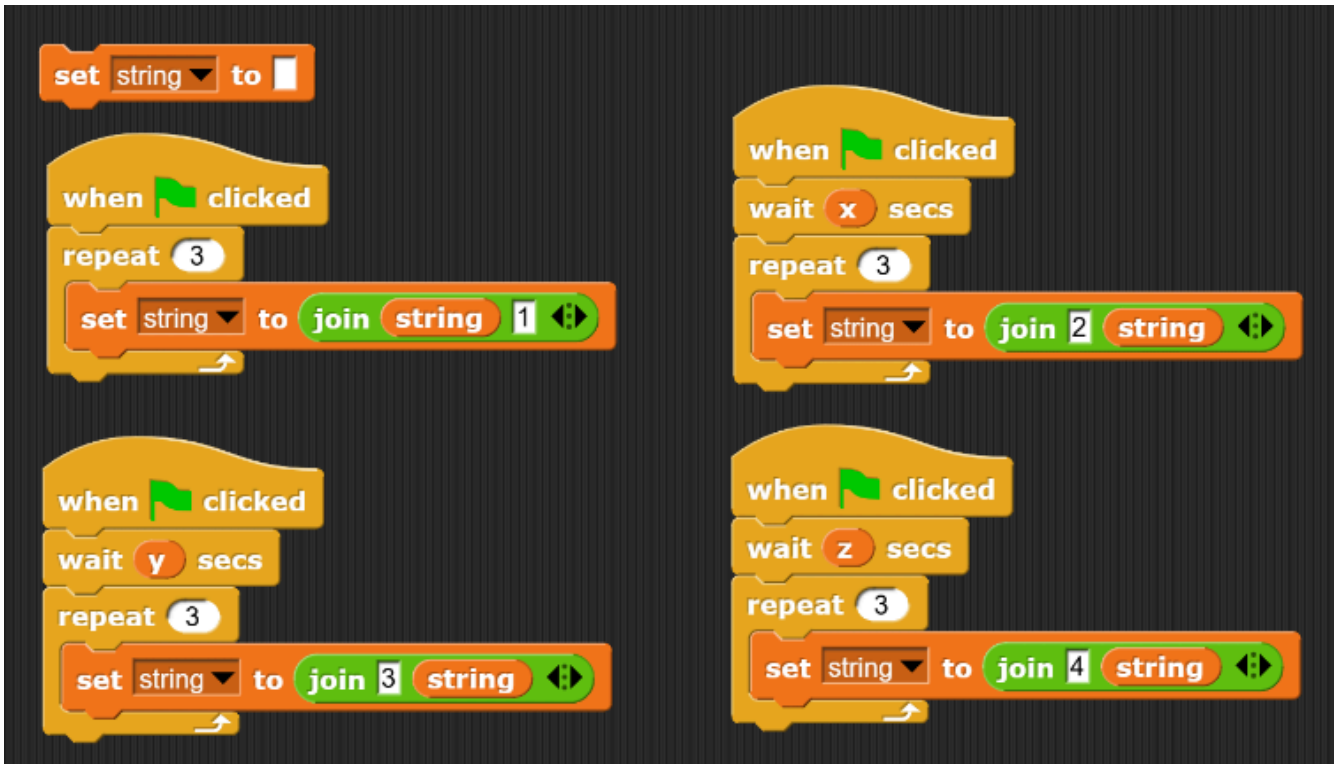
Amdahl's Law $S = \frac{1}{(1-P) + (\frac{P}{N})}$

1. You run a profiling program on a different program to find out what percent of time within the program each function takes. You get the following results:

Function	% Time
f	20%
g	50%
h	30%

   a. Assuming that each of these functions can be parallelized by the same speedup factor (N=2), which function, if parallelized, would cause the most speedup for the entire program? *Hint: Find the speedup (S) for each function.*

2. Assume you have the following script. What would string be once all scripts are done running for the given x, y, z values? If all scripts are running at the same time, you can assume the left topmost goes first. If x = 1, y = 1, and z = 1, the string would be “112341234234”



a. x = 1, y = 2, z = 2

b. x = 0, y = 4, z = 2

Optional Section (Extra Practice):

Optional Section I - Data Structures

---

1. To practice our Python, we're going to write a program that determines whether a year is a leap year or not. Let's call this function `is_it_a_leap_year()`, which takes in an integer `year` that represents the year we want to check for.

   To guide you in writing this problem, a leap year is defined as follows:

   1. If the year is perfectly divisible by 400, then it is a leap year, or
   2. If the year is perfectly divisible by 4 and it is not perfectly divisible by 100, then it is a leap year

   Write the function using an `if-elif-else` statement, and then think about how you would write it without using `elif`. Consider which implementation has much better readability.

```
def is_it_a_leap_year(year):
    _____
    _____
    _____
    _____
    _____
    _____
```

2. Determine the contents of `egg` in the following:

i.
```
yyyy = ["prep_disc", "prep_lab", "feed_alonzo"]
yyy = 8
yy = lambda y: len(y)

egg = [why[yyy] for why in yyyy if yy(why) > yyy]
```

ii.
```
numbers = [i for i in range(6)]
mutate = [numbers[i] * i for i in range(5)]
egg = numbers
```

iii.
```
egg = [x + 3 for x in [y - 2 for y in [z + 1 for z in range(6)]]]
```