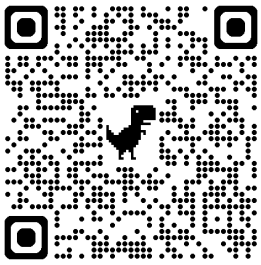
UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
Computer Science Division

CS10 Fall 2025 **TA: Victoria**



Discussion 11: Welcome to Python! 🐍

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 11" by next Discussion.
- For the worksheet, you can either explain the process in words, show a screenshot, or draw the block/process.
- Please complete the Feedback Form **tinyurl.com/fa25-disc-form**

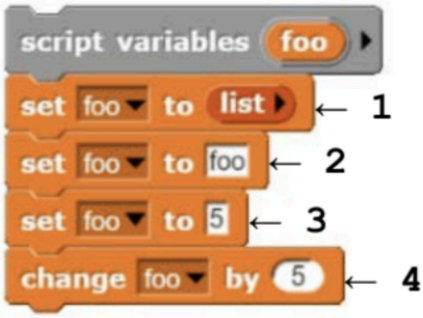
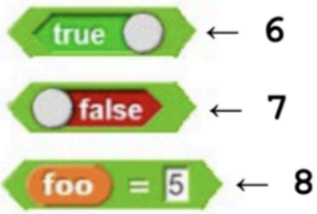
Group Activity / Question of the Day

- Ask the person to your right / left how they're doing! Midterm season can be stressful 🫠

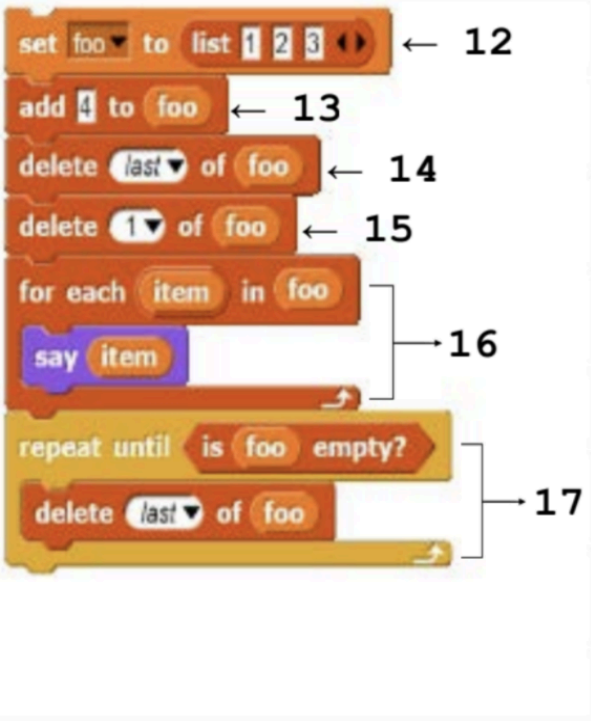
Required (Pages 2 - 7):

Section I - Snap! to Python

Translate the following Snap*!* code to its Python equivalent:

	1 2 3 4
	5a 5b
	6 7 8
	9 10 11

/



12

13

14

15

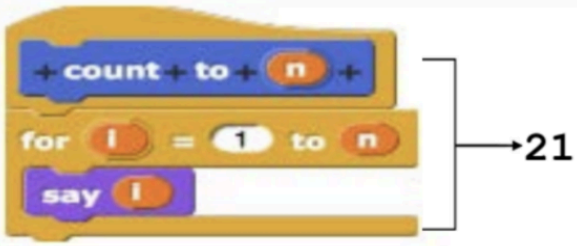
16

17

18

19

20



21

22

Section II - Write the Python Code

---

1. Write a program that determines whether a year is a leap 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. A leap year is defined as follows:
    a. If the year is divisible by 400, then it is a leap year, or
    b. If the year is divisible by 4 and not divisible by 100, then it is a leap year

```
def is_it_a_leap_year(year):
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
    _____________________________________________________
```

2. Fill in the blanks for the following function such that it satisfies its *docstring*. Credit: Professor John Denero

```
def remove(n, digit):
    """
    Return all digits of non-negative N that are not DIGIT,
    for some non-negative DIGIT less than 10.
    Examples:
    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept, digits = 0, 0
    while _____________________________________:
        n, last = n // 10, n % 10
        if _____________________________________:
            kept = _____________________________
            digits = ___________________________
    return _____________________________________
```

Section III - Understanding Python Functions

For each of the following functions, determine if they **Always** / **Never** / or **Sometimes** correctly find the maximum number in a **non-empty** list of numbers.

If you select always, leave the `example` boxes blank.
If you select never, give one example of a **3-element** list on which they fail, leaving the rest blank.If you select sometimes, give one example of a **3-element** list on which they work and one example of a **3-element** list on which they fail.

1.
```
def max1(values):
    sol = 0
    for v in values:
        if sol < v:
            sol = v
    return sol
```

Always: ○ Sometimes: ○ Never: ○

Example wherein it works: [ ]

Example wherein it doesn't work: [ ]

2.
```
def max2(values):
    smallest = min(values) # Assume `min` works.
    for v in values:
        if v > smallest:
            smallest = v
    return smallest
```

Always: ○ Sometimes: ○ Never: ○

Example wherein it works: [ ]

Example wherein it doesn't work: [ ]

3.
```
def max3(values):

    def helper(n, numbers):
        for number in numbers:
            if n < number:
                return False
        return True

    for value in values:
        if helper(value, values):
            return value
        else:
            return False
```

Always: ○ Sometimes: ○ Never: ○

Example wherein it works:

Example wherein it doesn't work:

Optional (Additional Practice):

Section I - Write the Python Code

---

1. Write a function `glorpify` that takes three arguments: num1, num2, and num3. It should return the result of multiplying num1 and num2, and adding num3 to the product.

2. Modify the function such that it now takes four arguments: num1, num2, num3, and a boolean argument: flag. If flag is True, it should return the result of multiplying num1 and num2, and adding num3 to the product. Otherwise, it should return the result of multiplying num1 and num2, and subtracting num3 to the product.

3. Modify the function such that it now takes five arguments: num1, num2, num3, flag, and a dyadic function g. If flag is True, it should return the result of calling g on num1 and num2, and adding num3 to the result. Otherwise, it should return the result of calling g on num1 and num2, and subtracting num3 from the result. *In part 2, g was "multiply."*

4. Write a function that accepts a list of digits: d, and prints the possible 3 digit numbers we can create using digits in d.

5. Now, modify the function such that it prints all the possible 3 digit numbers with *unique digits* that we can create using digits in d.

6. Now, modify the function such that it prints all the possible 3 digit numbers with digits arranged in strictly increasing order that we can create using the digits in d.

7. Now, modify the function such that it returns a **list** containing all the possible 3 digit numbers with *unique digits* that we can create using digits in d.