COMP110

CL17: Comparing Lists and Sets

Announcements

- EX04 (Dictionary Utils) due *Wednesday* (Oct 8)!
- Quiz 02 on Friday
  - Practice questions on the site
  - Review session on Thursday; details released, soon!
  - Please visit Office Hours for help!

Warm-Up:

Consider the following list. For each code sample below, write the corresponding output. Separate lines of output can be separated by a comma. If the code would raise an error, please write "error."

```
word: list[str] = ["C", "a", "t"]
```

0 1 2

generates a sequence from 0 up to, but not including, len(word)

7.3. What will be printed?

```
for x in range(0, len(word)):
  print(x)
```

3

0,1,2

0, 1, 2

7.6. What will be printed?

3

```
for x in range(1, len(word)):
  print(word[x])
```

1, 2

a, t

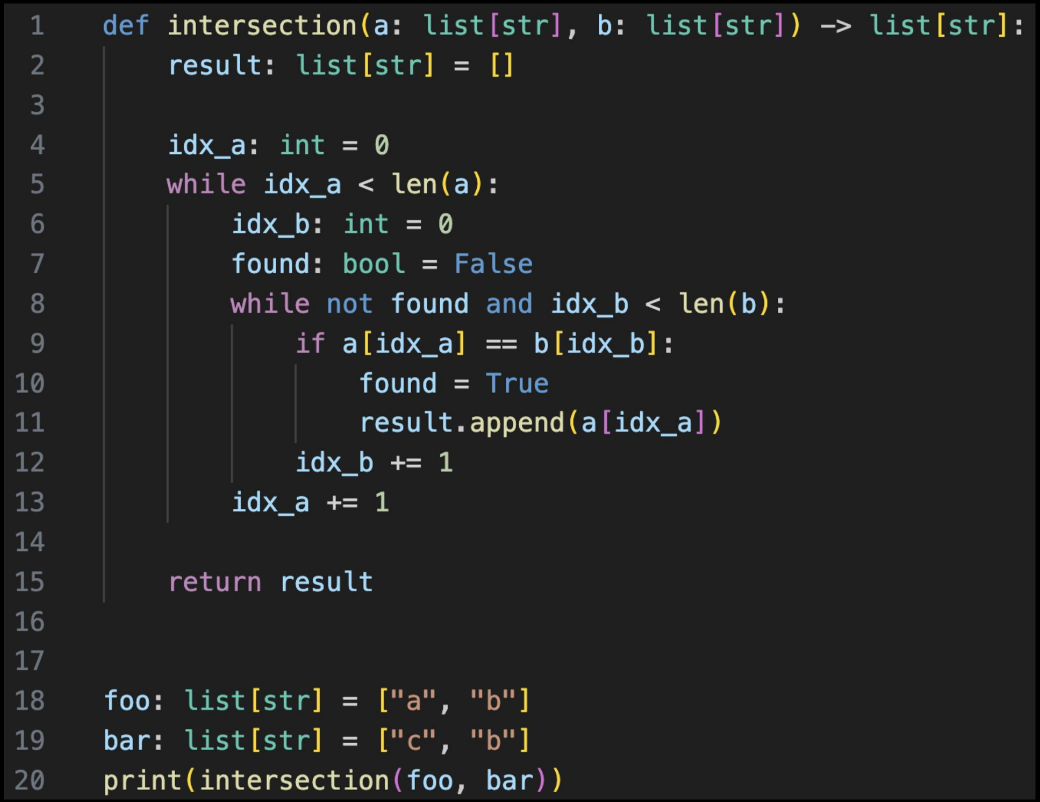
7.7. What will be printed?

```
for x in word:
  print(x)
```

← looping through the values in the list

C, a, t

With a neighbor, try diagramming:



```
1   def intersection(a: list[str], b: list[str]) -> list[str]:
2       result: list[str] = []
3   
4       idx_a: int = 0
5       while idx_a < len(a):
6           idx_b: int = 0
7           found: bool = False
8           while not found and idx_b < len(b):
9               if a[idx_a] == b[idx_b]:
10                  found = True
11                  result.append(a[idx_a])
12              idx_b += 1
13          idx_a += 1
14  
15      return result
16  
17  
18  foo: list[str] = ["a", "b"]
19  bar: list[str] = ["c", "b"]
20  print(intersection(foo, bar))
```

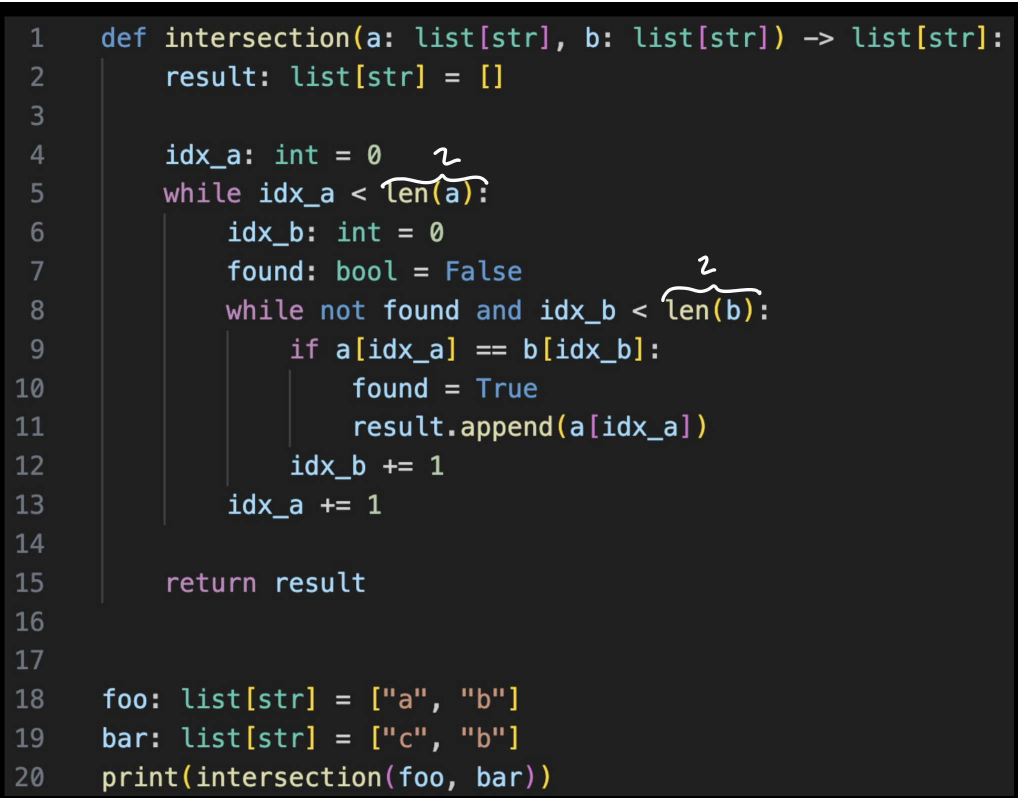
… and after diagramming:

Assume our unit of "operation" is the number of times the block of lines #9-12 are evaluated.

Q1. Can different values of a and b lead to a difference in the number of operations required for the intersection function evaluation to complete?

Q2. If so, provide example item values for a and b which require the fewest operations to complete? Then try for the maximal operations to complete?

Q3. Assuming the item values of a and b are random and unpredictable, about how many operations does this function take to complete?



```
1   def intersection(a: list[str], b: list[str]) -> list[str]:
2       result: list[str] = []
3
4       idx_a: int = 0
5       while idx_a < len(a):
6           idx_b: int = 0
7           found: bool = False
8           while not found and idx_b < len(b):
9               if a[idx_a] == b[idx_b]:
10                  found = True
11                  result.append(a[idx_a])
12              idx_b += 1
13          idx_a += 1
14
15      return result
16
17
18  foo: list[str] = ["a", "b"]
19  bar: list[str] = ["c", "b"]
20  print(intersection(foo, bar))
```

Output

["b"]

Stack

Globals

intersection	id:0
foo	id:1
bar	id:2

intersection

RA	20	a	id:1
RV	id:3	b	id:2
result	id:3	idx_a	~~0~~ ~~1~~ 2
found	~~False~~ True		
idx_b	~~0~~ ~~1~~ ~~2~~ ~~0~~ ~~1~~ 2		

Heap

id:0 fn lines 1-15

id:1 list[str]

0	"a"
1	"b"

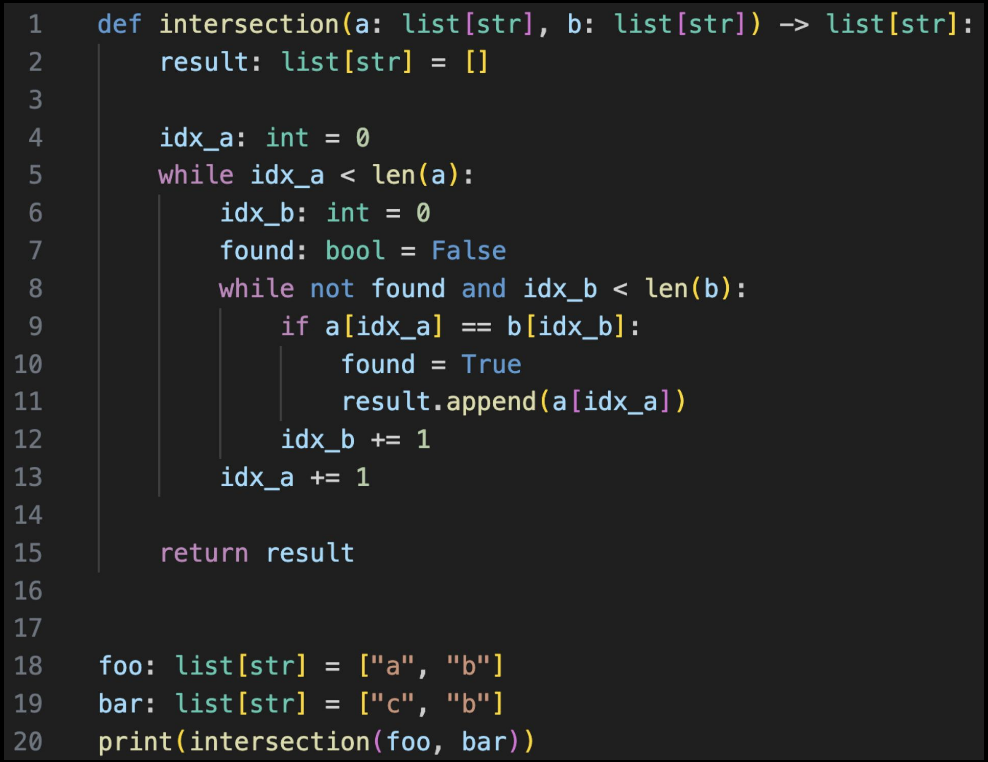
id:2 list[str]

0	"c"
1	"b"

id:3 list[str]

0	"b"

As the lengths of a and b grow, the number of operations grows *quadratically*



```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    idx_a: int = 0
    while idx_a < len(a):
        idx_b: int = 0
        found: bool = False
        while not found and idx_b < len(b):
            if a[idx_a] == b[idx_b]:
                found = True
                result.append(a[idx_a])
            idx_b += 1
        idx_a += 1

    return result


foo: list[str] = ["a", "b"]
bar: list[str] = ["c", "b"]
print(intersection(foo, bar))
```

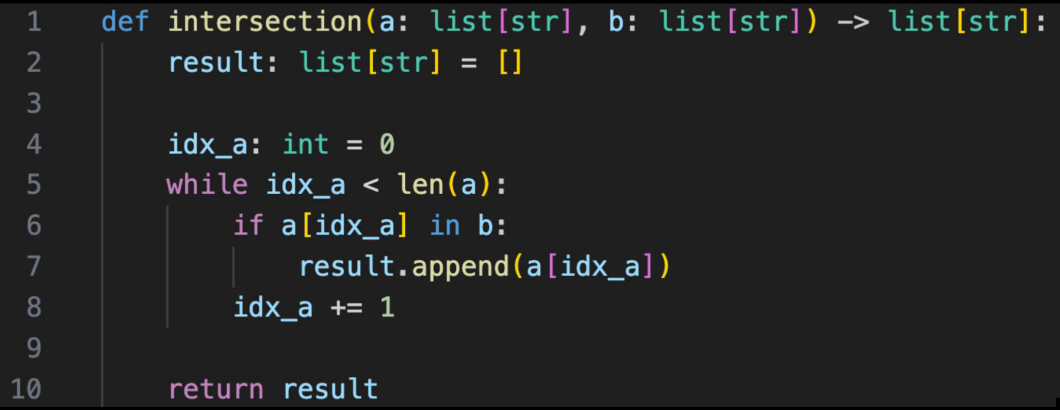
- Outer while loop iterates through each element of a
  - *If there are N elements, we'll iterate N times*
- And within each iteration of the outer while loop…
- The inner while loop iterates through elements of b until either:
  - We find a value that == the current element in a OR,
  - We have "visited" (accessed) every element in b
    - *If there are M elements in b, we'll iterate up to M times*

Assuming a and b both have 3 elements…

1. Example of values of a and b that will cause the ***fewest*** operations to occur?
   `intersection(a=["a", "a", "a"], b=["a", "b", "c"])`
2. Example of values of a and b that will cause the ***most*** operations to occur?
   `intersection(a=["a", "b", "c"], b=["d", "e", "f"])`

If list a has N elements and list b has M elements, the "worst case scenario" is that this code will cause N*M operations to occur.

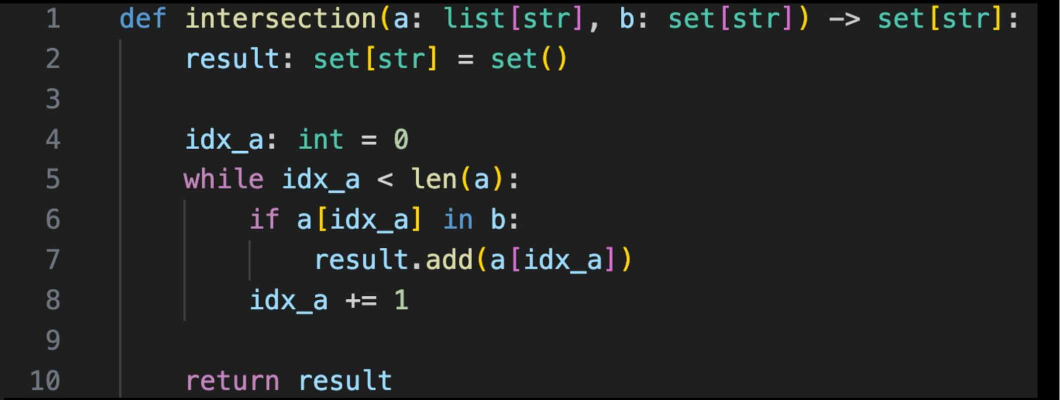
Comparing lists and sets



```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.append(a[idx_a])
        idx_a += 1

    return result
```



```python
def intersection(a: list[str], b: set[str]) -> set[str]:
    result: set[str] = set()

    idx_a: int = 0
    while idx_a < len(a):
        if a[idx_a] in b:
            result.add(a[idx_a])
        idx_a += 1

    return result
```

Suppose `a` and `b` each had 1,000,000 elements. The worst case difference here is approximately 1,000,000 operations, versus 1,000,000**2 or 1,000,000,000,000 operations.

If your device can perform 100,000,000 operations per second, then...

A call to `a` will complete in 2.78 hours and `b` will complete in 1/100th of a second.