Time Complexity 101 and Imports

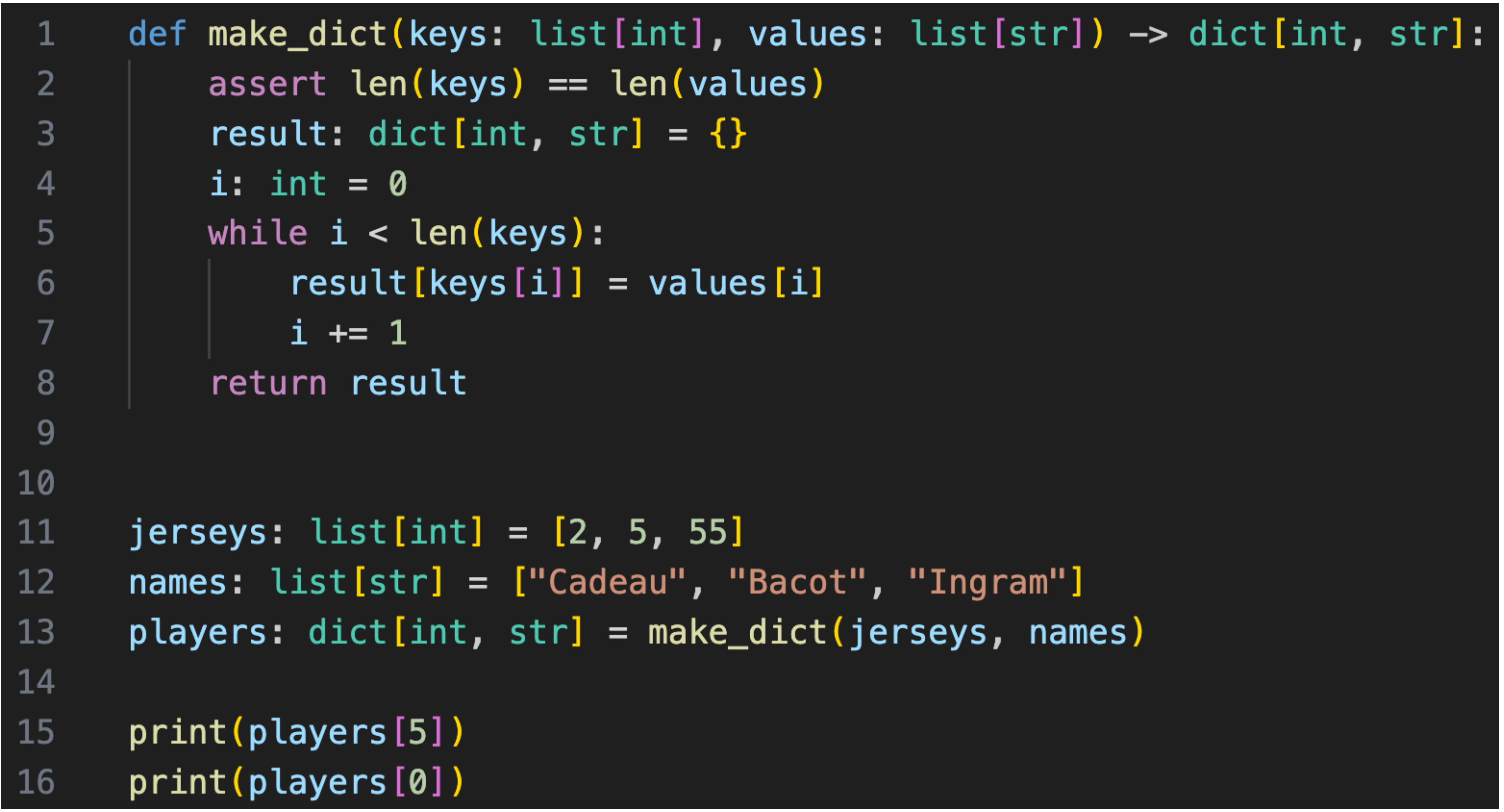
Today starts as a Paper + Pencil or Tablet + Pencil day... please keep laptops stowed away!

COMP110 - CL14

2024/03/21

Warm-up: Diagram this Code Listing

This will be today's attendance submission (CL14). Submit once complete.



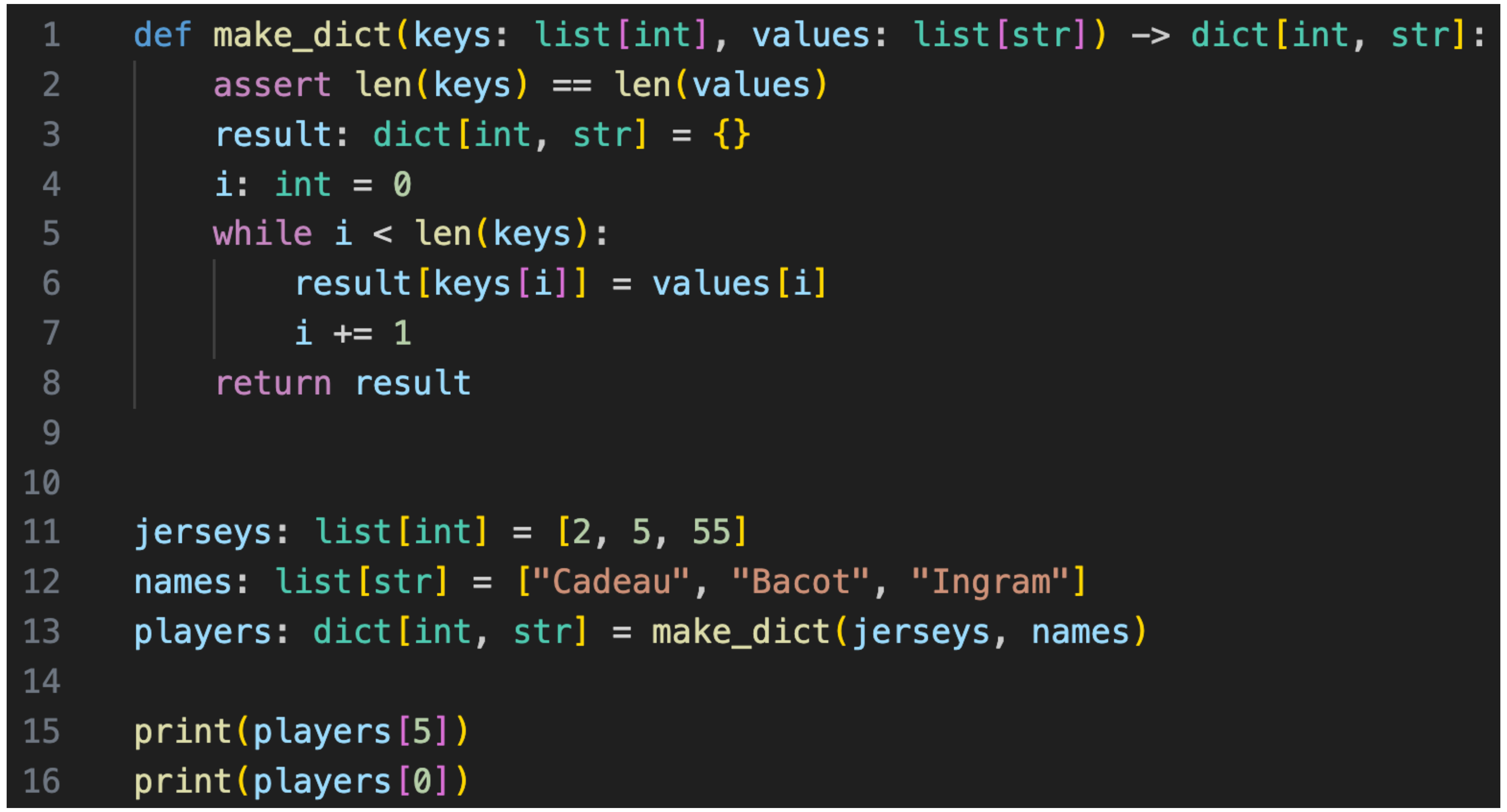
```python
def make_dict(keys: list[int], values: list[str]) -> dict[int, str]:
    assert len(keys) == len(values)
    result: dict[int, str] = {}
    i: int = 0
    while i < len(keys):
        result[keys[i]] = values[i]
        i += 1
    return result


jerseys: list[int] = [2, 5, 55]
names: list[str] = ["Cadeau", "Bacot", "Ingram"]
players: dict[int, str] = make_dict(jerseys, names)

print(players[5])
print(players[0])
```

Discussion question:

1. What *key* difference between dictionaries and lists is illustrated in the `players` variable's value?

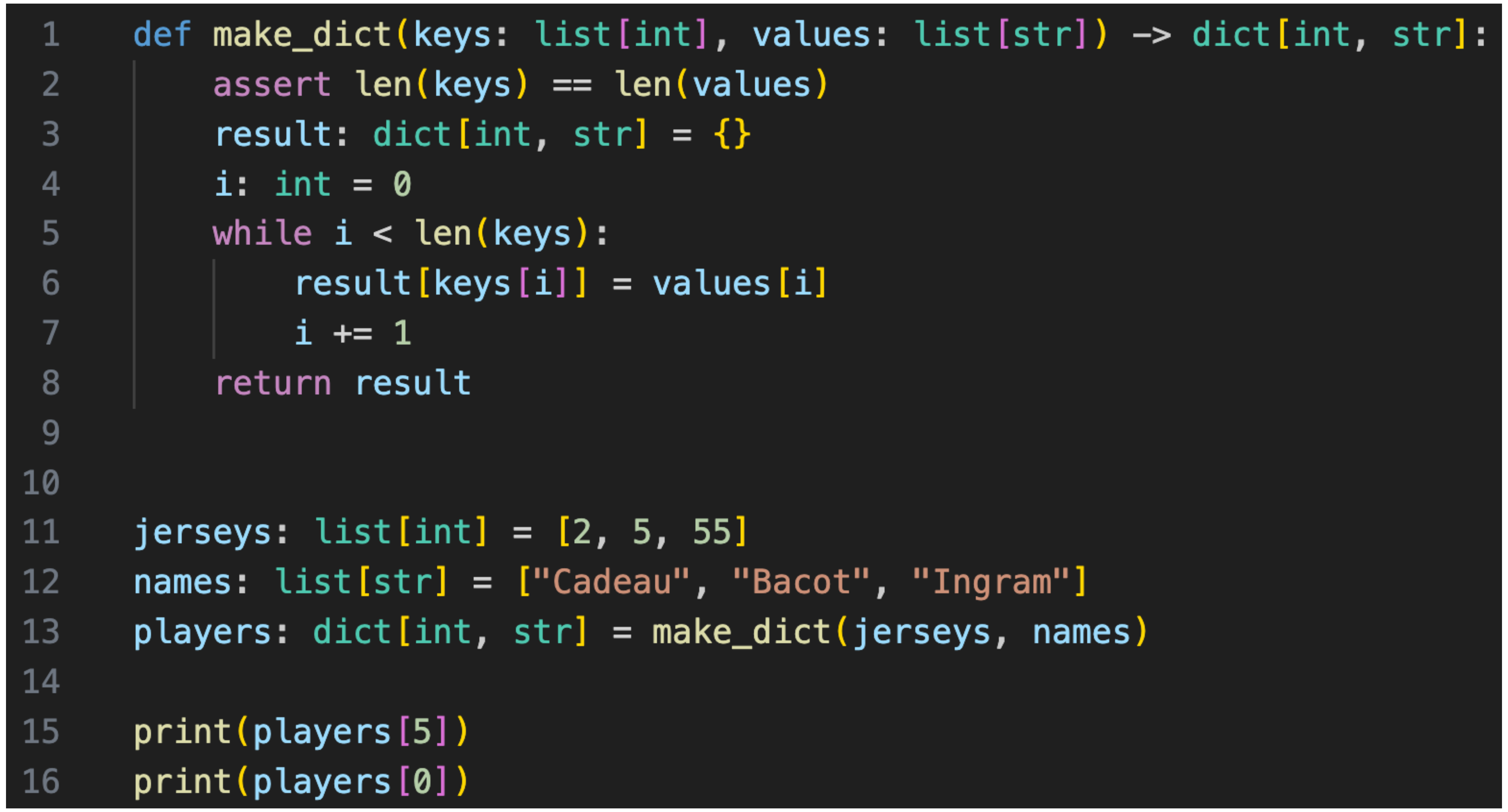


```python
def make_dict(keys: list[int], values: list[str]) -> dict[int, str]:
    assert len(keys) == len(values)
    result: dict[int, str] = {}
    i: int = 0
    while i < len(keys):
        result[keys[i]] = values[i]
        i += 1
    return result


jerseys: list[int] = [2, 5, 55]
names: list[str] = ["Cadeau", "Bacot", "Ingram"]
players: dict[int, str] = make_dict(jerseys, names)

print(players[5])
print(players[0])
```

Warm-up #2



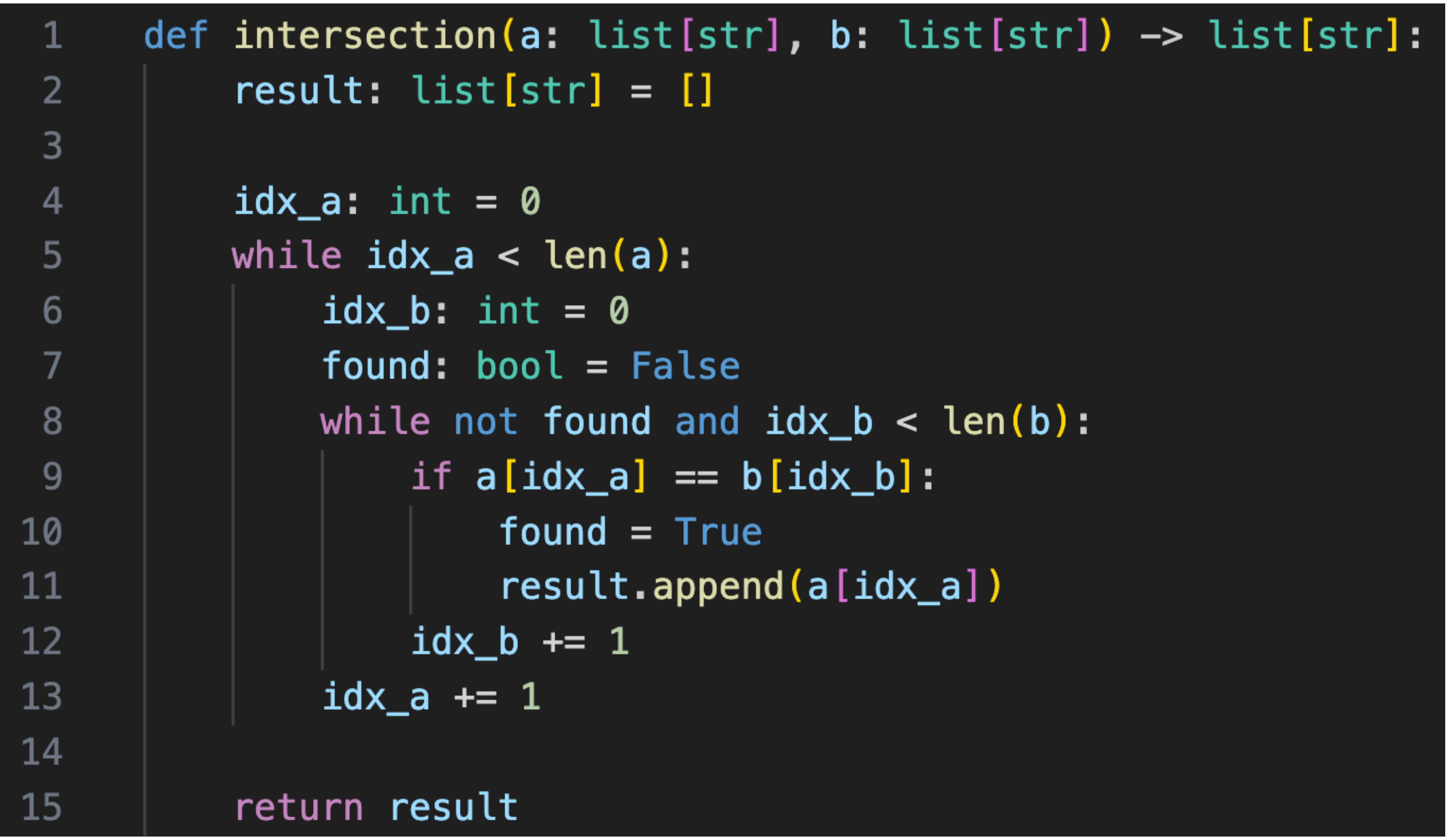
```
1   def make_dict(keys: list[int], values: list[str]) -> dict[int, str]:
2       assert len(keys) == len(values)
3       result: dict[int, str] = {}
4       i: int = 0
5       while i < len(keys):
6           result[keys[i]] = values[i]
7           i += 1
8       return result
9
10
11  jerseys: list[int] = [2, 5, 55]
12  names: list[str] = ["Cadeau", "Bacot", "Ingram"]
13  players: dict[int, str] = make_dict(jerseys, names)
14
15  print(players[5])
16  print(players[0])
```

Suppose keys and values each have 3 elements. Assume our unit of "operation" is the number of times the block of lines #6-7 are evaluated.

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

Q2. *How many* operations does this function take to complete in terms of N where N is len(keys)?

Warm-up #3: Given this example from Tuesday



```
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
```

Suppose a and b each have 3 elements. Assume our unit of "operation" is the number of times the block of lines #9-12 are evaluated.

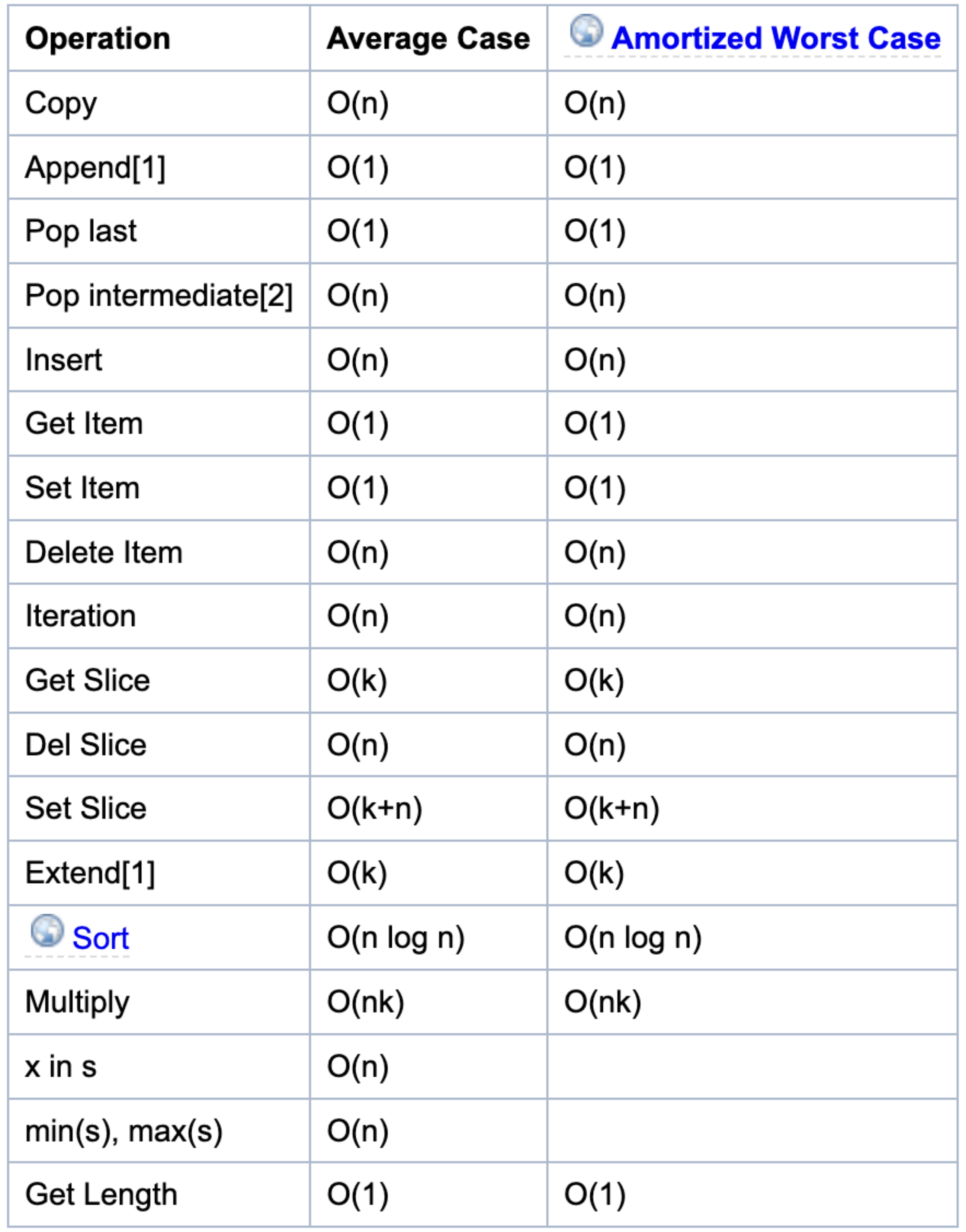
Q1. Can different values of *a* and *b* lead to a difference in the number of *operations* required for the *intersection* function evaluation to complete?
Q2. If so, provide example item values for *a* and *b* which require the *fewest* operations to complete? Then try for the *maximal* operations to complete?
Q3. Assuming the item values of a and b are random and unpredictable, *about how many* operations does this function take to complete?

Time Complexity of Common Operations of Fundamental Data Structures

Python's Built-in Capabilities Define Average and "Worst Case" Bounds for Common Data Structures

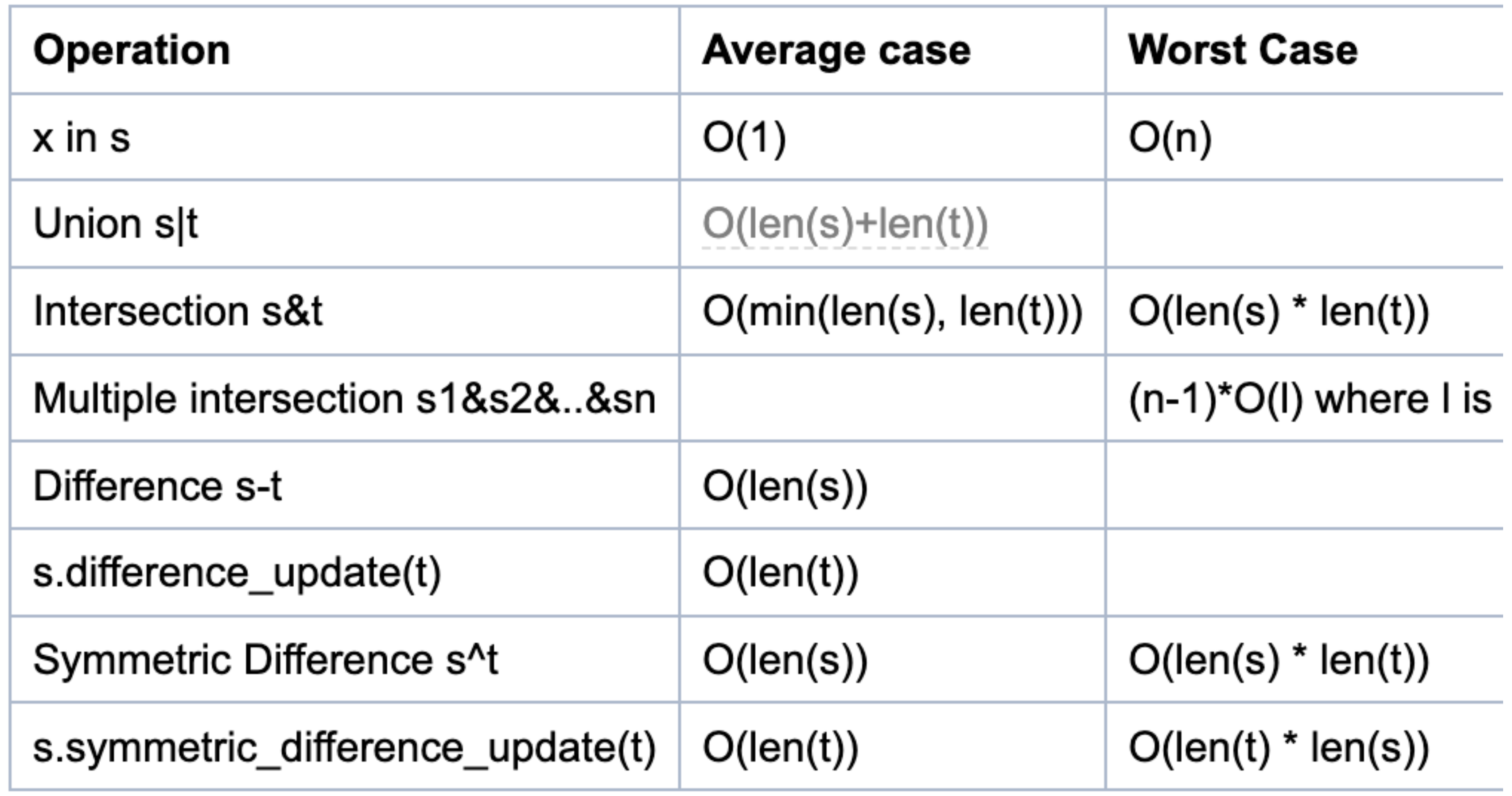
https://wiki.python.org/moin/TimeComplexity

list



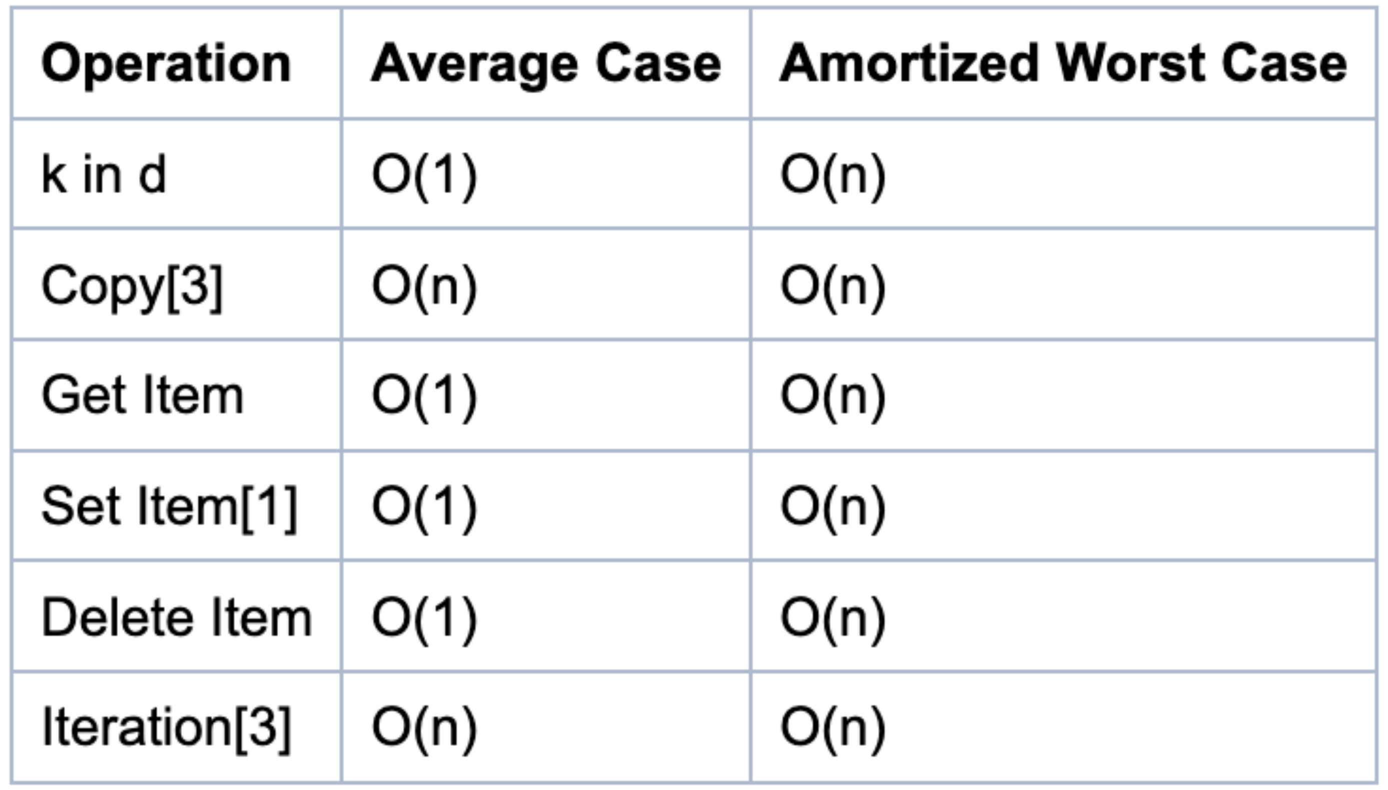
Operation	Average Case	Amortized Worst Case
Copy	O(n)	O(n)
Append[1]	O(1)	O(1)
Pop last	O(1)	O(1)
Pop intermediate[2]	O(n)	O(n)
Insert	O(n)	O(n)
Get Item	O(1)	O(1)
Set Item	O(1)	O(1)
Delete Item	O(n)	O(n)
Iteration	O(n)	O(n)
Get Slice	O(k)	O(k)
Del Slice	O(n)	O(n)
Set Slice	O(k+n)	O(k+n)
Extend[1]	O(k)	O(k)
Sort	O(n log n)	O(n log n)
Multiply	O(nk)	O(nk)
x in s	O(n)	
min(s), max(s)	O(n)	
Get Length	O(1)	O(1)

set



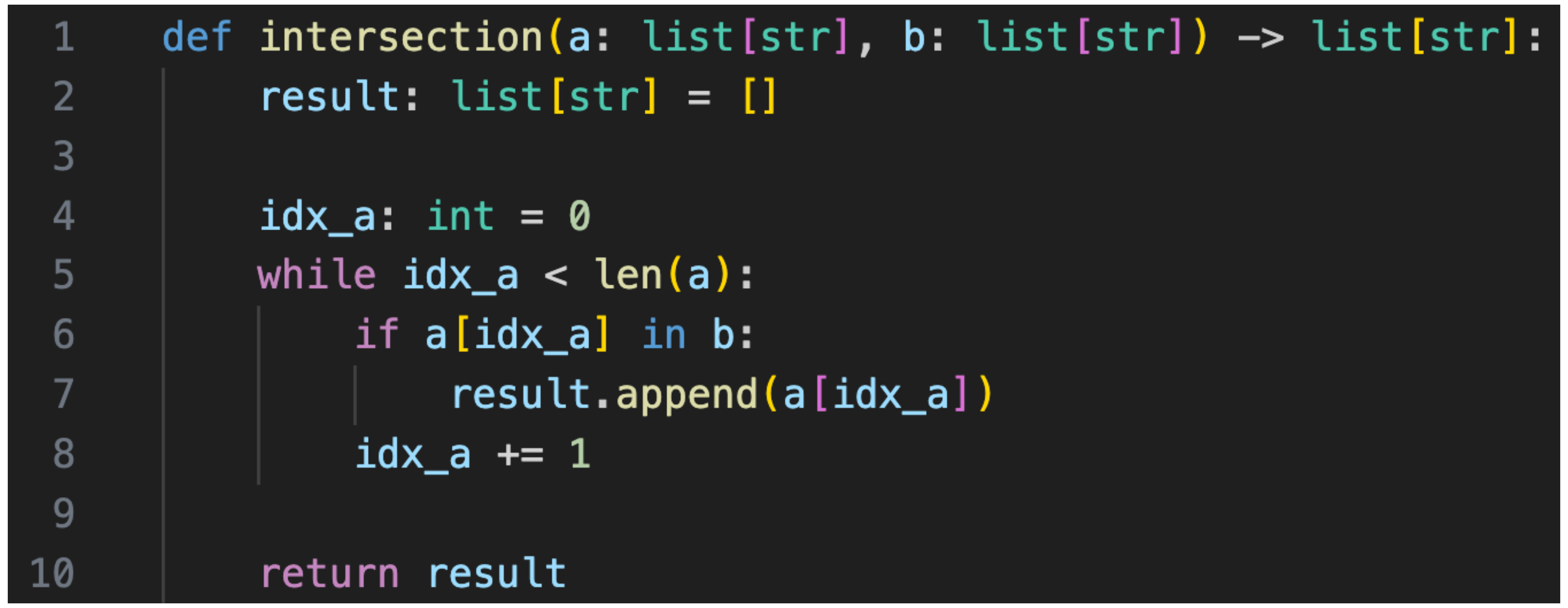
Operation	Average case	Worst Case
x in s	O(1)	O(n)
Union s|t	O(len(s)+len(t))	
Intersection s&t	O(min(len(s), len(t)))	O(len(s) * len(t))
Multiple intersection s1&s2&..&sn		(n-1)*O(l) where l is
Difference s-t	O(len(s))	
s.difference_update(t)	O(len(t))	
Symmetric Difference s^t	O(len(s))	O(len(s) * len(t))
s.symmetric_difference_update(t)	O(len(t))	O(len(t) * len(s))

dict



Operation	Average Case	Amortized Worst Case
k in d	O(1)	O(n)
Copy[3]	O(n)	O(n)
Get Item	O(1)	O(n)
Set Item[1]	O(1)	O(n)
Delete Item	O(1)	O(n)
Iteration[3]	O(n)	O(n)

Worst...

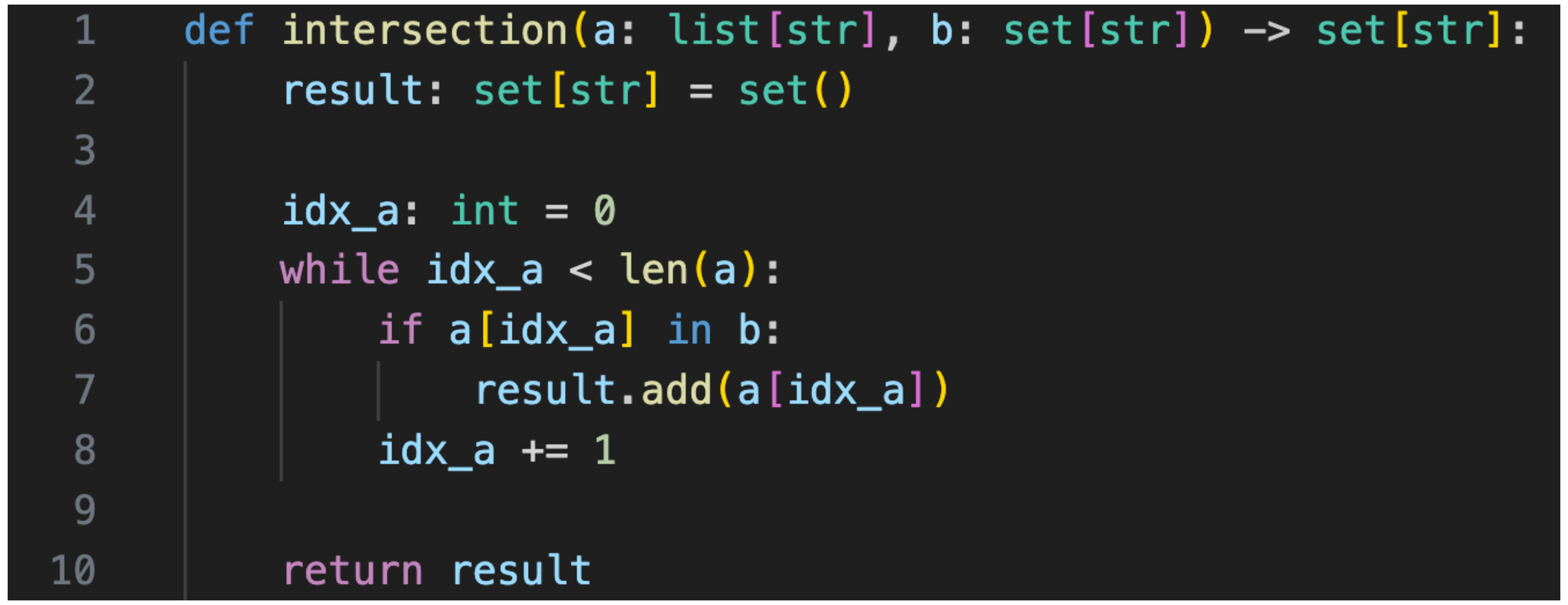


```python
def intersection(a: list[str], b: list[str]) -> list[str]:
    result: list[str] = []

    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
```

Orders of magnitude better...



```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**.

Imports (Code-along)

Functions as Parameters (Code Along)

Homework

- EX05 - Dictionary Utils - Due Monday 3/25 at 11:59pm
- RD00 - Ethical Algorithms - Due Friday 3/29 at 11:59pm