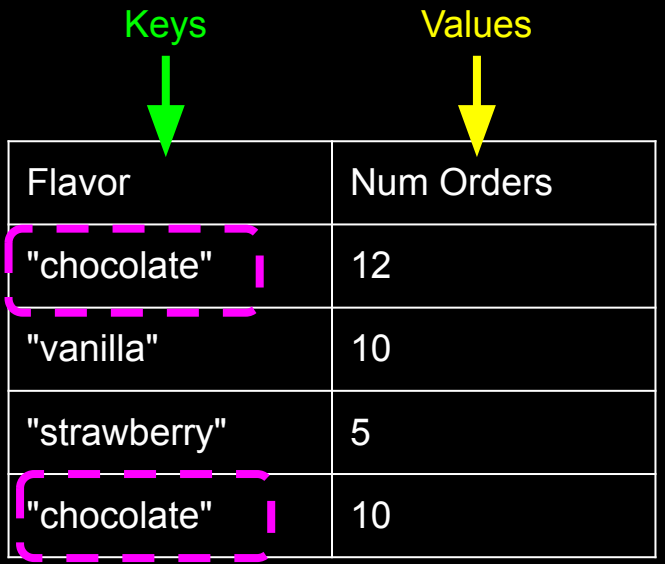
More on Sets and Dictionaries, and Intro to for Loops!

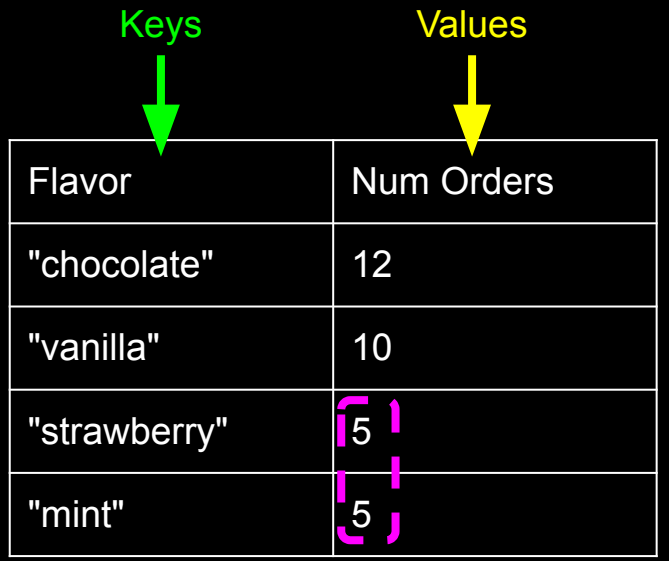
Announcements

- **LS11: Dictionaries** due tonight at 11:59pm
- **EX04: Dictionary Utils** will be posted to the website and Gradescope today, and due Tuesday, 10/7
- Today is your last day to submit regrade request(s) for Quiz 01!

Reminder: Can't Have Multiple of Same ***Key***



(Duplicate *values* are okay.)



Check if key in dictionary

<key> in <dict name>

"DC" in temps

"Florida" in temps

Let's try it!

Check if both the flavors "mint" and "chocolate" are in ice_cream.

Write a conditional that behaves the following way:
If "mint" is in ice_cream, print out how many orders of "mint" there are.
If it's not, print "no orders of mint".

Removing elements from a dict

Similar to lists, we use pop()

<dict name>.pop(<key>)

temps.pop("Florida")

Let's try it!

Remove the orders of "strawberry" from ice_cream.

`for` Loops

`for` loops iterate over the *keys* by default

```
for key in ice_cream:
    print(key)
```

```
for key in ice_cream:
    print(ice_cream[key])
```

Flavor	Num Orders
"chocolate"	12
"vanilla"	10
"strawberry"	5

Let's try it!

Use a for loop to print:
chocolate has 12 orders.
vanilla has 10 orders.
strawberry has 5 orders.

This is the code we wrote together, for reference.

```python
"""Examples of dictionary syntax with Ice Cream Shop order tallies."""

# Dictionary type is dict[key_type, value_type].
# Dictionary literals are curly brackets
# that surround with key:value pairs.
ice_cream: dict[str, int] = {
    "chocolate": 12,
    "vanilla": 8,
    "strawberry": 4,
}

# len evaluates to number of key-value entries
print(f"{len(ice_cream)} flavors")

# Add key-value entries using subscription notation
ice_cream["mint"] = 3

# Access values by their key using subscription
print(ice_cream["chocolate"])

# Re-assign values by their key using assignment
ice_cream["vanilla"] += 10

# Remove items by key using the pop method
ice_cream.pop("strawberry")

# Loop through items using for-in loops
total_orders: int = 0
# The variable (e.g. flavor) iterates over
# each key one-by-one in the dictionary.
for flavor in ice_cream:
    print(f"{flavor}: {ice_cream[flavor]}")
    total_orders += ice_cream[flavor]

print(f"Total orders: {total_orders}")
```

Sets!

Sets, like lists, are a *data structure* for storing collections of values.

Unlike lists, sets are *unordered* and each value has to be *unique*.
    Lists: *always* zero-based, sequential, integer indices!

Benefit of sets: testing for the existence of an item takes only one “operation,” regardless of the set’s size.

```
pids: set[int] = {730120710, 730234567, 730000000}
```

To add a value to the set:

```
pids.add(730123456)  # Add a value to the set
```

To remove a value from the set:

```
pids.remove(730120710)  # Remove a value from the set
```

Data structure	Allows duplicates?	Ordered?	Fast lookups?	Use Case
list [ ]	✅	✅	❌	Ordered collections
set {}	❌	❌	✅	Unique values, membership testing (fast lookups)
dictionary {key: value}	✅ (duplicate values allowed; keys must be unique!)	It's complicated	✅	Mappings, fast lookups, counting

Match the Data Structure to its Application

Set

List

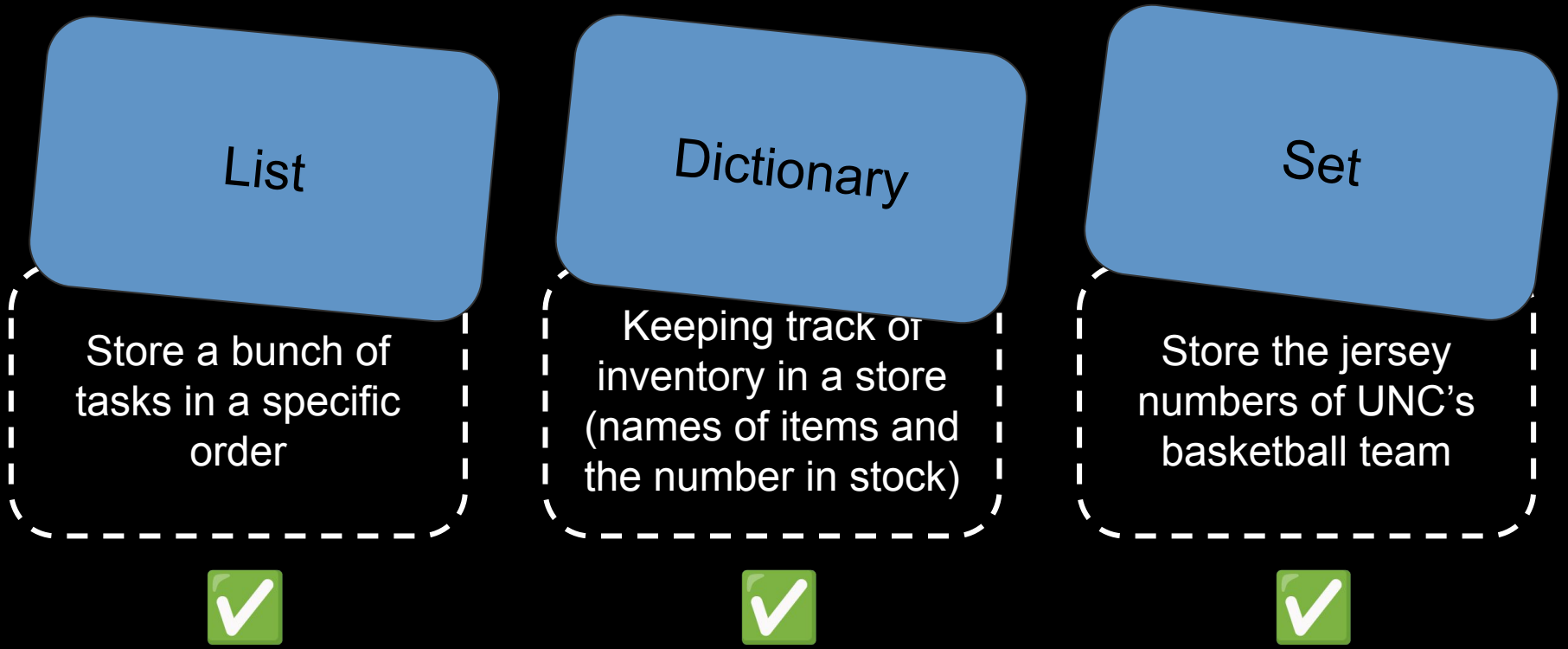
Dictionary

Store a bunch of tasks in a specific order

Keeping track of inventory in a store (names of items and the number in stock)

Store the jersey numbers of UNC's basketball team

Match the Data Structure to its Application



```
1 vend: dict[str,str] = {"A1":"Oreos", "A2":"Lays", "B1":"Coke", "B2":"7up"}
2 flavors: set[str] = {"Orange", "Cherry", "Lime"}
```

2.1. What will be printed?

```
1 for prod in vend:
2   print(prod)
```

2.2. What will be printed?

```
1 for prod in vend:
2   print(vend[prod])
```

2.3. What will be printed?

```
1 for flav in flavors:
2   print(flav)
```

2.4. What will be printed?

```
1 if "Berry" in flavors:
2   print("Available!")
3 else:
4   print("Out...")
```

2.5. What will be printed?

```
1 def buy(vm: dict[str,str])->str:
2   for thing in vm:
3     return thing
4   return "Other"
5
6 print(buy(vm=vend))
```

Memory Diagram

```python
def group_names(names: list[str]) -> dict[str, int]:
    groups: dict[str, int] = {}
    first_letter: str
    for n in names:
        first_letter = n[0]
        if first_letter in groups:
            groups[first_letter] += 1
        else:
            groups[first_letter] = 1
    return groups

ppl: list[str] = ["Karen", "Emily", "Kris"]
output: dict[str, int] = group_names(names=ppl)
print(output)
output["I"] = 1
print(output)
```