Question 1: Multiple Choice For each of the next questions, select all of `set`, `list`, and/or `dict` for which the statement describes. Bubble in ALL squares that apply.

1.1. Which of the following data structures are sequences?
■ `list` ☐ `set` ☐ `dict`

1.2. Select all data structures that are mutable.
■ `list` ■ `set` ■ `dict`

1.3. Select all data structures that can contain duplicate values.
■ `list` ☐ `set` ■ `dict`

1.4. Which of these data structures use key-value pairs for storing data?
☐ `list` ☐ `set` ■ `dict`

1.5. Which of the following data structures does not guarantee the order of elements? (The `dict` data structure is intentionally omitted; in Python, order is maintained. However, generally, `dict`-like data structures do not guarantee ordering.)
☐ `list` ■ `set`

1.6. Which data structures allow indexing via subscription notation to access individual elements directly?
■ `list` ☐ `set` ■ `dict`

1.7. If you need to store a collection of items and frequently check whether an item is in the collection, which data structure is most efficient?
☐ `list` ■ `set` ☐ `dict`

1.8. To ensure the order of elements is maintained and allow for duplicates, which data structure would you choose?
■ `list` ☐ `set` ☐ `dict`

1.9. Which of the following data structures require the `.add()` method to add a value?
☐ `list` ■ `set` ☐ `dict`

1.10. To store a sequence of elements that you intend to iterate over and modify, which data structure offers the best performance?
■ `list` ☐ `set` ☐ `dict`

1.11. For associating student PIDs to their respective email addresses, which data structure provides the most efficient lookup?
☐ `list` ☐ `set` ■ `dict`

1.12. Which data structure's *literal syntax* is enclosed within curly braces?
☐ `list` ■ `set` ■ `dict`

1.13. Which data structure's *literal syntax* is enclosed within square brackets?
■ `list` ☐ `set` ☐ `dict`

1.14. Which data structures can you iterate over using a `for..in` loop?
■ `list` ■ `set` ■ `dict`

1.15. Which data structures allow the use of the `len` function to determine the *number of elements* it contains?
■ `list` ■ `set` ■ `dict`

1.16. Which of the following data structures is best when you want to find the *intersection*, *union*, or *difference* between two collections of values?
☐ `list` ■ `set` ☐ `dict`

1.17. If you were creating a messaging app, where you want to maintain a list of messages in the order they were received, which data structure would you use?
■ `list` ☐ `set` ☐ `dict`

1.18. When trying to count the frequency of words in a document, which data structure would allow you to efficiently store and update counts?
☐ `list` ☐ `set` ■ `dict`

1.19. If you want to specify the data type with which a collection of values is indexed, which data structure should you use?
☐ `list` ☐ `set` ■ `dict`

Question 2: Looping Short Answer Consider the following dictionary and set. 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."

```
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)
```

Solution: `A1, A2, B1, B2`

2.2. What will be printed?

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

Solution: `Oreos, Lays, Coke, 7up`

2.3. What will be printed?

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

Solution: `Cherry, Lime, Orange`

2.4. What will be printed?

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

Solution: `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))
```

Solution: `A1`

Question 3: Respond to the following questions. Consider the following dictionary:

```
1 vend: dict[str,str] = {"A1":"Oreos", "A2":"Lays", "B1":"Coke", "B2":"7up"}
```

3.1. Write a line of code to find the length of the `vend` dictionary.

Solution: `len(vend)`

3.2. Write a line of code to add the key-value pair, "B3" and "Fanta", to the dictionary.

Solution: `vend["B3"] = "Fanta"`

3.3. Write a line of code to change the value associated with the key, "A1", to "Twix".

Solution: `vend["A1"] = "Twix"`

3.4. Write a line of code to remove the key-value pair, "A2" and "Lays", from the dictionary.

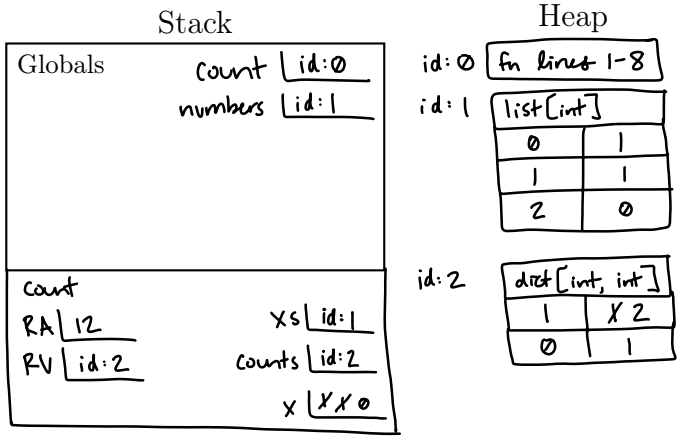
Solution: `vend.pop("A2")`

Question 4: Memory Diagram Trace a memory diagram of the following code listing.

```
1  def count(xs: list[int]) -> dict[int, int]:
2    counts: dict[int, int] = {}
3    for x in xs:
4      if x in counts:
5        counts[x] += 1
6      else:
7        counts[x] = 1
8    return counts
9
10
11 numbers: list[int] = [1, 1, 0]
12 print(count(numbers))
```

Output

```
{1:2, 0:1}
```

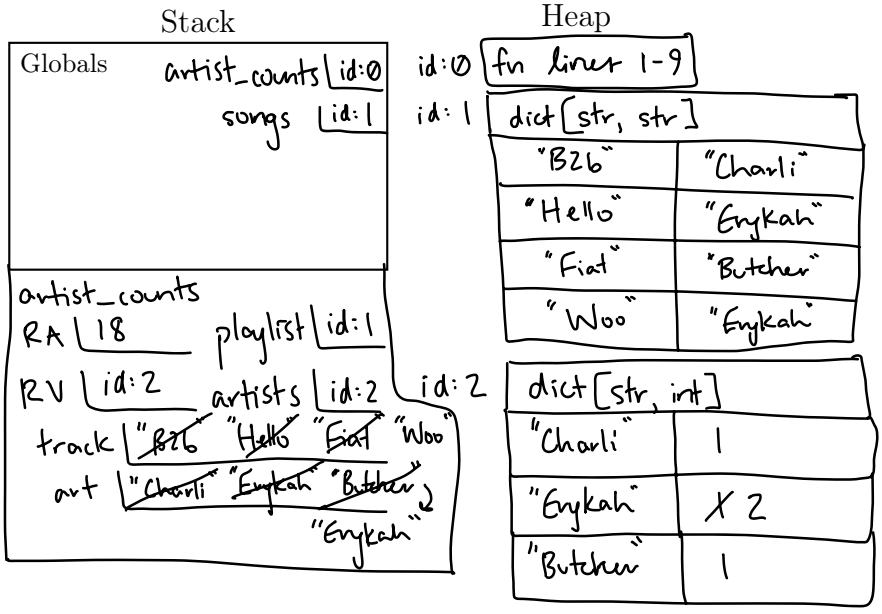


Question 5: Memory Diagram Trace a memory diagram of the following code listing.

```
1  def artist_counts(playlist: dict[str, str]) -> dict[str, int]:
2    artists: dict[str, int] = dict()
3  • for track in playlist:
4      art: str = playlist[track]
5      if playlist[track] not in artists:
6        artists[art] = 1
7      else:
8        artists[art] += 1
9    return artists
10
11 songs: dict[str, str] = {
12   "B2b": "Charli",
13   "Hello": "Erykah",
14   "Fiat": "Butcher",
15   "Woo": "Erykah"
16 }
17
18 print(artist_counts(songs))
```

Output

```
{"Charli": 1, "Erykah": 2, "Butcher": 1}
```



Question 6: Function Writing Write a function definition for `count_lens` with the following expectations:

- The `count_lens` function should accept a list of string values and return a dictionary where the key type is `int` and the value type is `int`.
- The function should *count the frequencies* of strings in the parameter list of the same length(s). For example, `["a", "b", "cc", "d"]` should return `{1: 3, 2: 1}` because there were three strings of length 1 and one string of length 2.
- You should explicitly type all variables, parameters, and return types.

6.1. Write your function definition for `count_lens` here.

Solution: One possible solution, of many possible valid solutions:

```
def count_lens(strs: list[str]) -> dict[int, int]:
  counts: dict[int, int] = {}
  for s in strs:
    if len(s) in counts:
      counts[len(s)] += 1
    else:
      counts[len(s)] = 1
  return counts

```

6.2. Write a test function for a use case that demonstrates expected usage with at least three values in the list. Your input should be different from the prompt's sample input.

Solution: One possible test function, of many possible valid test functions:

```
def test_count_lens() -> None:
  """Test flip flop with 5 elements"""
  letters: list[str] = [["a", "b", "cc", "dd"]
  assert count_lens(letters) == {1: 2, 2: 2}
```

Question 7: EXTRA: Want more practice with loops? 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."

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

7.1. What will be printed?

```
1 i: int = 0
2 while i < len(word):
3   print(word[i])
4   i += 1
```

Solution: C, a, t

7.2. What will be printed?

```
1 def grab(val: list[str]) -> str:
2   i: int = 1
3   while i < len(val):
4     return val[i]
5     i += 1
6
7 print(grab(word))
```

Solution: a

7.3. What will be printed?

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

Solution: 0, 1, 2

7.4. What will be printed?

```
1 for x in word:
2   print(word[x])
```

Solution: TypeError (or just error)

(We haven't learned range yet!)

7.5. What will be printed?

```
1 i: int = 0
2 while i < len(word):
3   print(word[i])
4   i += 1
```

Solution: C, a, t

7.6. What will be printed?

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

Solution: a, t

7.7. What will be printed?

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

Solution: C, a, t

7.8. What will be printed?

```
1 i: int = 0
2 while i < (len(word) - 1):
3   print(i)
4   i += 1
```

Solution: 0, 1