# EXERCISE 1.2

1. Imagine you’re having a conversation with a future colleague about whether to use the iPython Shell instead of Python’s default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?

Compared to Python, IPython can do everything that python can do. Ipython provides even extra features like tab completion, testing, debugging, system calls and many other features.

2. Python has a host of different data types that allow you to store and organize information. List 4 examples of data types that Python recognizes, briefly define them, and indicate whether they are scalar or non-scalar.

| **Data type** | **Definition** | **Scalar or Non-Scalar?** |
|---|---|---|
| Int | Represents both positive and negative integers | Scalar |
| Float | Represents both positive and negative decimals | Scalar |
| Lists | A mutable ordered sequence in Python | Non-scalar |
| Strings | A mutable array of characters in Python | Non-scalar |

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.

A list is mutable so it can be changed. A tuple is immutable so it can not be changed.

4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you’re creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.

For this app, | would create a dictionary for each vocabulary word that would follow this format:

Word = {‘Definition’: ‘insert definition’, ‘Part of Speech’: ‘insert part of speech’}

I would use a dictionary for each word because it would allow me to organize the content into key-value pairs. This is the best way to display both the definition and part of speech in one structure. After creating all of the words, | would put them in a list called vocabulary_words like so:

vocabulary_words = [ ‘word’, ‘word’, ‘word’…]

I would use a list because that would allow the user to add new words and remove old ones when they are finished.