Introduction to Dictionaries

1. Introduction

Python dictionaries are a built-in data type used to store collections of key-value pairs. They are unordered (prior to Python 3.7), mutable, and highly optimized for retrieving data. In the context of data science, dictionaries are indispensable for managing structured data, configuration settings, and mapping operations.

2. Syntax and Basic Usage

A dictionary is defined using curly braces {} with keys and values separated by colons:

```
student = {
    "name": "Oscar",
    "age": 33,
    "courses": ["Math", "Stats"]
}
print(student["name"])  # Output: Oscar
```

3. Applications in Data Science

Dictionaries are widely used in various data science workflows:

3.1 Configuration Management

Used to store model parameters, data paths, and options in training pipelines.

```
params = {
    "learning_rate": 0.01,
    "batch_size": 32,
    "epochs": 50
}
```

3.2 Data Mapping

Helpful in replacing or mapping categorical variables.

```
gender_map = {"M": 0, "F": 1}
df["gender_mapped"] = df["gender"].map(gender_map)
```

3.3 Aggregation Results

Store intermediate results from groupings or computations.

```
results = {
    "mean": df["sales"].mean(),
    "std": df["sales"].std(),
    "min": df["sales"].min(),
    "max": df["sales"].max()
}
```

3.4 Feature Importance and Model Metrics

Dictionaries are ideal for storing evaluation metrics across models.

```
metrics = {
    "model_1": {"accuracy": 0.93, "f1": 0.91},
    "model_2": {"accuracy": 0.89, "f1": 0.87}
}
```

4. Pros of Using Dictionaries

- **Fast Lookups:** Access time is near constant due to hash-based indexing.
- **Readable:** Key-value pairs make code self-documenting.
- **Flexible:** Can store complex data like lists, dictionaries, or objects as values.
- **Built-in Methods:** Includes useful methods like `.get()`, `.items()`, `.update()`.

5. Limitations of Dictionaries

- **Key Constraints:** Keys must be hashable (immutable).

- **Memory Overhead:** More memory usage compared to lists.
- **No Vectorized Operations:** Not suited for numerical operations (use `numpy` or `pandas` for that).

6. Best Practices

- Use dictionaries for mapping and quick retrieval, not for numerical computations.
- Use comprehensions for concise initialization:

```
squares = {x: x**2 for x in range(5)}
```

- Use nested dictionaries for hierarchical information (but beware of complexity).

7. Conclusion

Dictionaries are one of Python's most powerful and versatile data structures. For data scientists, they provide a clean and efficient way to manage metadata, configuration, and mappings in models and pipelines. Mastering dictionaries is a foundational step in writing clean and maintainable Python code in data science workflows.