Lesson 4 – Data Structures: Organizing Options

Why Data Structures?

When solving optimization problems, we usually have **sets of options**: routes, items, or decisions. Instead of making one variable for each option, we can store them together in a structure like a **list** or **dictionary**.

Optimization Connection

Data structures allow us to **store many possibilities**, then use loops and functions to test and compare them. This is how we scale from one decision to many.

Lists: Keeping Options in Order

A **list** is an ordered collection of items. We use square brackets `[]` to create them.
Example: Route times

```
times = [18, 13, 16, 11]
print(times[0]) # first element
print(times[-1]) # last element
```

Output

```
18
11
```

Looping over a list

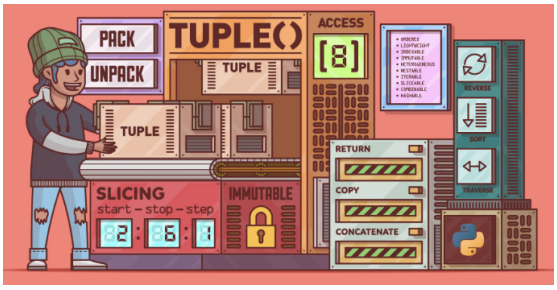
```
for t in times:
    print("Route takes", t, "minutes")
```

Output

```
Route takes 18 minutes
Route takes 13 minutes
Route takes 16 minutes
Route takes 11 minutes
```

Tuples: Fixed Pairs of Data

A **tuple** groups multiple values into one item, written with `()`. This is useful for options with multiple attributes (time, fun).



Example: Routes with time and fun

```
routes = [(12, 8), (15, 10), (9, 6)]
for r in routes:
    print("Time:", r[0], "Fun:", r[1])
```

Output

```
Time: 12 Fun: 8
Time: 15 Fun: 10
Time: 9 Fun: 6
```

Dictionaries: Naming the Options

A **dictionary** stores pairs of `key:value`. We can look up values by name.
Example: Snack prices

```
prices = {"apple": 2, "banana": 1, "chips": 3}
print(prices["apple"])
print(prices["chips"])
```

Output

```
2
3
```

Image: upload healthyjunk.png (arrows apple->2, banana->1, chips->3)

Looping through a dictionary

```
for snack, cost in prices.items():
    print(snack, "costs", cost)
```

Output

```
apple costs 2
banana costs 1
chips costs 3
```

Optimization Connection

With dictionaries we can store options by name and retrieve their values quickly. This mirrors real-world decision problems where items have labels (jobs, machines, routes).

Lesson 4 Activities

1. Create a list of 5 numbers and print the largest using a loop.
2. Make a list of 3 prices, then use a loop to print only those under `budget=10`.
3. Store 3 routes as tuples `(time, fun)` and print the one with the best fun score.
4. Make a dictionary of 3 snacks with their prices and print the cheapest one.
5. **Challenge:** From a dictionary of `(price, fun)` pairs, pick the most fun item within `budget=12`.