IS1111 Tutorial 5 – **Dictionaries (More Advanced)**

1) Nested Dictionaries – Access, Update & Safe Lookup

We store employee data in a nested dictionary:

```
employees = {
  "E001": {"name": "Sarah", "dept": "Sales", "salary": 45000},
  "E002": {"name": "Cian", "dept": "IT", "salary": 52000},
  "E003": {"name": "Aoife", "dept": "Sales", "salary": 48000},
  "E004": {"name": "Ben", "dept": "IT", "salary": 47000},
}
```

a) Accessing Nested Data

Print the **name** of employee "E002".

b) Updating Nested Data

Increase Ben's salary by 3000.
Then print Ben's updated salary.

c) Safe Lookup

If an employee ID does not exist, print "Employee not found" instead of crashing.
Test this using "E999".

d) Writing a Function

Write a function:

def get_salary(employees, emp_id):

It should:

- return the salary if the employee exists
- return None if the employee does not exist

Test it using:

- "E001"
- "E999"

---

2) Grouping & Aggregation

Using the same employees dictionary:

a)

Create a dictionary called dept_summary that stores:

- department name as the key
- value as another dictionary with:
  - "count" (number of employees)
  - "total_salary"

Example structure:

```
{
  "Sales": {"count": 2, "total_salary":
93000},
  "IT": {"count": 2, "total_salary": 99000}
```

b)

Print each department and its total salary.

c)

Find which department has the highest total salary.

---

3) Dictionary ↔ List Conversions

```
scores = {
  "Alice": 88,
  "Brian": 75,
  "Ciara": 92,
  "David": 75,
}
```

a)

Convert the dictionary into a list of tuples using .items().

b)

Sort the students by score (highest first).

c)

Create a new dictionary where:

- keys are the same names
- values are "Pass" if score ≥ 80
- otherwise "Fail"

(You may use a dictionary comprehension.)

---

4) ID Generation & Filtering

```
transactions = {
  "T001": {"amount": 250.0, "type": "deposit"},
  "T002": {"amount": 100.0, "type": "withdrawal"},
  "T003": {"amount": 400.0, "type": "deposit"},
}
```

a)

Write a function:

def generate_transaction_id(transactions):

It should:

- find the highest transaction number
- return the next ID in format "T004"

b)

Add a new transaction:

- amount: 150.0
- type: "deposit"