CoGrammar

Week 5 – Open Class 3

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Software Engineering Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Objectives

1. List & Dictionary functionality.
2. Key operations relevant to Lists & Dictionaries.
3. List & Dictionary application for improved data management.
4. Open floor Q&A

Dictionaries

- ★ In Python, dictionaries function akin to the dictionaries we commonly used in English class, such as those from Oxford.
- ★ Python dictionaries are similar to a list, however each item has two parts, a key and a value.
- ★ To draw a parallel, consider an English dictionary where the key represents a word, and the associated value is its definition.

Dictionary Example

```
# Dictionary Example

my_dictionary = {
    "name": "Terry",
    "age": 24,
    "is_funny": False
}
```

Dictionaries are enclosed in curly brackets; key value pairs are separated by colon and each pair is separated by a comma. On the left is the key, on the right is the value.

Dict. Functions

- The dict() function in Python is a versatile way to create dictionaries.
- Create dictionaries through assigning values to keys by passing in keys and values separated by an **=** sign.

```
# Creating a dictionary with direct key-value pairs
my_dict = dict(name="Kitty", age=25, city="Belarus")
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus'}
```

Dict Update

- To append or add elements to a dictionary in Python, you can use the update() method or simply use the square bracket notation.

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict.update({'breed': 'Shorthair'})
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict['breed'] = 'Shorthair'
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

Dictionary Cont.

- To access a value in a dictionary, we simply call the key and Python will return the value paired with said key.
- Similar to indexing, however we provide a key name instead of an index number.

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
name = my_dict["name"]
# Output: 'Kitty'
```

Dictionary Operations

- **Key-Value Pairs**
  - items()
- **Fetching**
  - get()
- **Updating**
  - update()
- **Deleting**
  - pop() / popitem()

Wrapping Up

Lists

2D lists in Python offer a powerful mechanism for organizing and manipulating data in a structured manner.

Dictionaries

Rows represent individual lists within the main list, while columns denote elements within each of these lists.

CoGrammar

Questions around Operations on Lists and Dictionaries

CoGrammar

Thank you for joining