Week 6: COMP-801 - Integrated Computing Practice

Agenda

- Dictionaries
  - AR5 dictioanry exercises
- Python objects, object identity
- Getting started on Lab 4
  - Test, design, and implement the constructor in class

`dict` : Built-in Dictionary Datat Type

Repesents a collection of **key**, **value** pairs

- Values in the dictionaries are organized by their keys
- The **keys** are ***immutable*** objects
- The **values** are any type of object

Dictionary Operations

- **Indexing []**
  - access a value in the dictionary based on the key associated with that value
- **Item assignment**
  - Include a new key, value pair in a dictionary
  - Change the value of an existing pair based on the key
- **del** operator
  - Remove a pair based on the key

Dictionry Methods

- items(), values(), keys()
- get()
- pop(), popitem()
- clear()
- update()

Dictionry Practice

- How to define a dictionary object
- How to access dictionary elements, keys, values
- How to modify dictionary elements and values

Objects in Python

In Python EVERYTHING developers define is an object

- literals are objects, whether they represent numeric values, Boolean values, string values
- function definitions are objects
- method definitions are objects
- EVEN a class definition is an object

Object Identity in Python

An **object** in Python has

- **identity**: unique number to locate the object in the program memory
- **value**: data and characteristics the object holds

A **variable** is a name corresponding to the object identity

- Convenience for developers to refer to an object by name
- INSTEAD of using the identity (a long number)

Objects in Python - Example

- Define names to represent variables, functions, classes, ...
- Inspect their: data type, value, and identity

```
def detailed_type(object):
    return (type(object), object, id(object))

result = detailed_type("Hello, World")
print(result)
```

Lab4: Understand and Document the Project

- Write proper module docstrings in ALL five `.py` files
- Enter developer and collaborator(s) names in `DESIGN.md`
- Version control this development step.

Lab4: Experiment with the Project

- Are there any **Problems** reported by VS Code?
- Are there Python errors when you run the `.py` modules?

Lab4: Trace Project Execution with the Debugger

- Launch the debugger
- Set a breakpoint where `main()` is called in `client.py`
- Run the debugger to trace the execution and understand the code

Lab4: Test the Constructor

- Write a 2nd testing function for the constructor
- Run, debug, fix errors, and resolve any VS Code proglames
- Version control this step

Lab4: Design the Constructor

- Discuss the design of the constructor with the whole class
- Write the design with your partner
- Verify the design by "tracing" the execution "on paper"
- Version control this step

Lab4: Implement the Constructor

- Use the design to write the implementation
- Run, debug, fix errors, and resolve any VS Code proglames
- Version control this step