

FYI, Part 2 covered arrays which will not be included in the required part of 217.

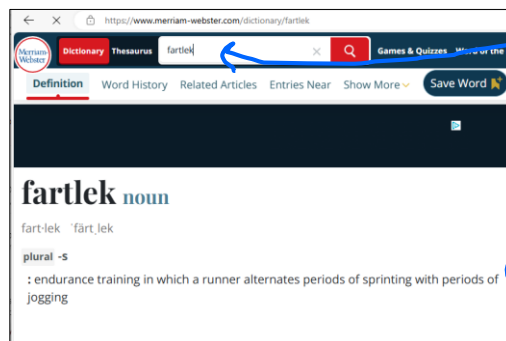
## Composites, Dictionaries: Part 3

A composite type that employs a key and data value pair to allow for more efficient retrievals of elements.

James Tam

### Real Life Dictionary

- When using a dictionary one “looks up” the word in the dictionary.
- The word provides the search parameter in order to retrieve additional details.



Lookup key

Lookup value

James Tam

## Real Life: Other Cases Of Looking Values Using A Key

- Workers: using the Social Insurance Number (key) to find additional information about the person (data value).
- Students: using the student identification number (key) to find student information such as: courses, grades, contact information... (data values).
- ISBN: the unique identifier for a book (key) to find details about the book such as: title, author, publisher... (data values).

James Tam

## Python Dictionaries

- Similar to lists, strings and arrays, a dictionary is a composite type that can be decomposed into multiple parts.
- The curly braces specifies a dictionary.
- Each element needs a pair
  - Key : data
    - Key: an integer that uniquely identifies the element (don't try to repeat keys).
    - Data: the information that is stored for the element.
  - Example: **111: Jim Tam, undeclared**
    - 111: the lookup key (in this case the student identification number)
    - Jim Tam, undeclared: the data value for the element (in this case it's the information about the student).
- Individual elements are accessed via key **not as a index** from zero to (size-1).

James Tam

## Creating A Dictionary

- **Format:**

```
<dictionary name> = {<key1>:<value1>, <key2>:<value2>,  
    <key3>:<value3>,... <key^n>:<value^n>}
```

- **Example:**

```
students = {111:"James Tam,business,computer science",  
            250:"Jim Tam(Ace student,undeclared",  
            117:"Scott Bruce,business"}
```

```
aDictionary = {}    #A reference to an empty dictionary
```

- **Important details:**

- Key: needs to be one: integer, bool, string, float – avoid the latter even though it meets syntax requirements because of precision issues).
  - Keys must be unique, repeating a key overwrites the previous value.
- Values aren't limited.

James Tam

## Accessing Dictionary Elements

- **Single element**

- **Format:**

```
<dictionary>[<key>]
```

- **Example:**

```
print(students[studentID])
```

- **All elements**

- **Format:**

```
for <key> in <dictionary>:  
    students[<key>]
```

- **Example:**

```
for aStudentID in students:  
    print(aStudentID,students[aStudentID])
```

James Tam

## Dictionary Basics: Complete Example

- Name of the complete online example:

1dictionary\_basics.py

```
students = {111:"James Tam,business,computer science",
            250:"Jim Tam(Ace student,undeclared",
            117:"Scott Bruce,business"}

studentID = 111
print(students[studentID])

print("Number of students %d" %(len(students)))

for aStudentID in students: #aStudentID=key
    print("\t%d: %s" %(aStudentID,students[aStudentID]))
```

Unless the keys are a fixed ascending/descending sequence a while-loop can't be used. Dictionary values are retrieved through the keys not through an index from zero to (size-1)

James Tam

## Other Dictionary Operations

- Many list operations are possible with dictionaries
- Review of operations you just saw:
  - Creating a new dictionary
  - Access single element
  - Stepping through all elements
    - Getting a key for an element (needed for an assignment)
- Changing elements (to be covered in the next example)
  1. Add/creating new element (no such key exists).
  2. Update/modifying an existing element (key already exists).
  3. Creating a dictionary dynamically via a loop (needed for an assignment).
  4. Deleting elements.
  5. Checking for membership in the dictionary.
  6. Clearing the dictionary (deleting the whole thing).

```
for aStudentID in students:
    print("\t%d: %s" %(aStudentID,
```

```
#3rd operation: Create dictionary dynamically via a loop
while(i <= size):
    scientists[i] = "Scientist #" + str(i)
    i = i + 1
```

James Tam

## Other Dictionary Operations: Complete Example

- Name of the full online example:  
2creating\_modifying\_elements.py

```
scientists = {}
i = 1
size = 4
#3rd operation: Create dictionary dynamically via a loop
while(i <= size):
    scientists[i] = "Scientist #" + str(i)
    i = i + 1

size = size + 1
#1st operation: Add/create new element
scientists[size] = "Scientist #" + str(size)

#2nd operation: Change/modify existing element (key = 2)
scientists[2] = "James Tam"
```

James Tam

## Other Dictionary Operations: Complete Example (2)

```
#New dictionary operation #4: Deleting element by key
del scientists[1]

#5th dictionary operation: Checking for membership
key = 1
if key in scientists:
    print("Scientist with ID %d exists" %(key))
else:
    print("No such scientist with ID %d" %(key))
key = 2
if key in scientists:
    print("Scientist with ID %d exists" %(key))
else:
    print("No such scientist with ID %d" %(key))

#6th dictionary operation: Clearing the whole dictionary
scientists.clear()
```

James Tam

## Dictionary Key-Related Functions

| Operation | Examples (Zhao/Hudson)               | Description (students=dictionary)                |
|-----------|--------------------------------------|--------------------------------------------------|
| List      | <code>list(students.keys())</code>   | List all keys in dictionary                      |
| Sort      | <code>sorted(students.keys())</code> | Sort the dictionary by the keys                  |
| Keys      | <code>students.keys()</code>         | Get all the keys in the dictionary               |
| Items     | <code>students.items()</code>        | Get all the items (key:values) in the dictionary |

### Functions already covered

| Operation      | Examples where it was covered                                      |
|----------------|--------------------------------------------------------------------|
| Access via key | <code>print("\t%d: %s" %(aStudentID, students[aStudentID]))</code> |
| Membership     | <code>if key in scientists:</code>                                 |
| Length         | <code>print("Number of students %d" %(len(students)))</code>       |
| Clear          | <code>scientists.clear()</code>                                    |
| Add / Create   | <code>scientists[size] = "Scientist #" + str(size)</code>          |
| Delete         | <code>del scientists[1]</code>                                     |

James Tam

## When Can Dictionaries Be Used?

- Composite data needs to be represented but one field uniquely identifies the element (it becomes the key).
  - Aforementioned examples: SIN, student identification number etc.
- The key is how the dictionary element is retrieved.

James Tam

## Why Use Dictionaries Over Lists?

- Efficiency (quickly) of accessing elements.
  - You will learn more about evaluating efficiency of your programs in CPSC 331 and in CPSC 413.
- To quickly access list elements it's typically sorted.
- But the sorting operation takes time.
  - And time is still required to access elements with a sorted list.
- Dictionaries use the programming technique called 'hashing' to allow for a fast lookup of the data values.
  - You will learn details about hashing algorithms in CPSC 331 but briefly the hashing maps the key to an index via the hashing function.
  - Bottom line: rather than the lookup time be affected by the size of the list (without hashing) the lookup time is a constant value.

James Tam

## Implementation Of Dictionaries In Memory

- Similar to lists, creating a dictionary allocates a reference in memory.
  - These are references to an empty list and an empty dictionary respectively.  
`aList = []`  
`aDictionary = {}`
- Consequently "passing a dictionary" as a function parameter is actually passing a reference to the dictionary (address of the dictionary).

James Tam

## **After This Section You Should Now Know**

- How to create a new dictionary using the methods covered.
- Valid data types for the key and data value parts of a dictionary element.
- How to access keys and data values using the methods covered.
- How to iterate through the elements in a dictionary.
- Other common operation: adding new elements, checking for membership, clearing dictionary, deleting elements, modifying existing elements.
- How changes to dictionaries made inside a function will change the actual dictionary because references are employed.
- When and why to use dictionaries

James Tam

## **Copyright Notification**

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.

slide 16

James Tam