**CHAPTER 9 – DICTIONARIES – KEY POINTS**

* A dictionary is another **aggregate** data type.
* It’s a collection of **key : value pairs**. Values are **indexed** by their keys.
* A **colon** is required **between** each key and value.
* The key **:** value pairs are separated by commas within **curly braces**.

**Examples**

A dictionary of math test scores with strings as keys and integers as values:

**math = {'Test 1':88,'Test 2':81,'Test 3':77,'Test 4':90}**

A dictionary of Olympic running race results with integer keys and strings as values:

**podium = {1:'USA',2:'Canada',3:'Spain'}**

A dictionary of wedding vips with strings for keys and strings for values:

**vips = {'bride':'Penny','groom':'Kenny','maid':'Jenny'}**

* The **values** in a dictionary can be of **any type**. Keys must be **immutable**.
* You can make a **dictionary of lists** and even a **dictionary of dictionaries**!
* Dictionary values can be retrieved by specifying the key inside **square brackets**.

**gold\_medal = podium[1] # assigns 'USA' to gold\_medal**

**low\_math = math['Test 3'] # assigns 77 to low\_math**

* The **in** and **not in** operators can be used to detect if a **key** is in the dictionary.
* Elements can be added (or re-assigned) with a key inside square brackets.

**vips['best man'] = 'Benny'**

**math['Test 5'] = 86**

* The **del** statement can delete an element by a specified key inside square brackets.
* The **len** function returns the number of elements in a dictionary.
* Page 473 shows that both values and key types can be mixed up. This is rarely needed.
* An empty dictionary can be created by leaving the curly braces empty (also on page 473).
* A for loop can be used to iterate **over the keys** in a dictionary.

**for k in vips:**

**print(k, end = ' ')**

* The loop above prints: bride groom maid best man

**for k in vips:**

**print(f'{vips[k]} is the {k}')**

* But, you can use an **f-string** to display the value, too. The loop above prints:

Penny is the bride

Kenny is the groom

Jenny is the maid

Benny is the best man

* The important Table 9-1 on page 475 shows some **dictionary methods**.
* NOTE: to use a method, precede it with the dictionary **name and the dot operat**or.
* The methods are explained very well on pages 475-480.
* The **items() method** is especially useful for iterating over a dictionary.
* items() returns a special sequence called a **dictionary view**.

**for k,v in vips.items():**

**print(f'{v} is the {k}') # output is same as previous loop**

* The loop above uses k and v to represent key and value, but any variable names work.
* Pages 480-489 have some good sample programs that employ dictionaries.

**Dictionary Comprehensions** (page 490-493)

* You can use this awesome capability to make dictionaries, too.
* **Figure 9-1** at the bottom of page 490 shows the parts needed to code one.

**Examples**

A dictionary with integers from 2 to 10 as keys and their squares as values.

**squares = {x:x \*\* 2 for x in range(2,11)}**

The dictionary created is:

**{2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}**

You can add an **if condition** if desired.

A dictionary with **even** integers from 2 to 10 as keys and their squares as values.

**squares = {x:x \*\* 2 for x in range(2,11) if x % 2 == 0}**

The dictionary created is:

**{2: 4, 4: 16, 6: 36, 8: 64, 10: 100}**

NOTE: The chapter starts in on **SETS** on page 493. **Sets are not in the course**.