10. DICTIONARIES

You will all have used a dictionary at some point to find an explanation or a definition of a word. You look up the word and find the matching explanation or definition.

We can also use a similar data structure in Python. We have already looked at lists as a versatile data structure, where we can add to a list, reverse a list, count the items in a list, check the list index, etc.

The main difference between a list and a dictionary is the use of **keys** to access data items in the dictionary, not their index position. We create a dictionary by creating matching pairs of a key and a value that is linked to the key.

Example:

```
national_dish = {'Austria':'Weiner Schnitzel',
                 'Belgium':'Moules-frites',
                 'Poland':'Kotlet schabowy',
                 'Sweden':'Kottbullar',
                 'Germany':'Currywurst',
                 'Greece':'Moussaka'}

print(national_dish['Belgium'])
```

```
>>>
Moules-frites
>>>
```

As you can see, I can access the value by using the relevant key, using the dictionary name and the key in square brackets. The keys and the values in a dictionary can be anything; the point is that the key is ***mapped to*** the value so we can search for the value by referencing the key.

If I try to search for a key that is not in my dictionary, this will raise an error. You will need to make sure that you add code to deal with that error in your program.

```
print(national_dish['United Kingdom'])
```

```
>>>
Traceback (most recent call last):
  File "C:/Users/Example Files/DictionaryExample1.py",
line 10, in <module>
    print(national_dish['United Kingdom'])
KeyError: 'United Kingdom'
>>>
```


One way to do this is to use the 'in' operator to check:

```
if "UK" in national_dish:
    print("The national dish of {0} is {1}".format("UK",national_dish["UK"]))
else:
    print("Key value not in dictionary")
```

```
>>>
Key value not in dictionary
>>>
```

We can access all the keys and values in the dictionary by using a **for loop** to iterate over them and print them out:

```
print("Country\tNational Dish\n")
for key, value in(national_dish.items()):
    print("{0}\t{1}".format(key,value))
```

```
>>>
Country National Dish

Poland  Kotlet schabowy
Belgium Moules-frites
Sweden  Kottbullar
Germany Currywurst
Greece  Moussaka
Austria Weiner Schnitzel
>>>
```

The dictionary is not sorted; we can use the built-in function to produce a sorted dictionary (by key) and iterate over both the keys and the corresponding values.

```
print("Country\tNational Dish\n")
for key, value in sorted(national_dish.items()):
    print("{0}\t{1}".format(key,value))
```

```
>>>
Country National Dish

Austria Weiner Schnitzel
Belgium Moules-frites
Germany Currywurst
Greece  Moussaka
Poland  Kotlet schabowy
Sweden  Kottbullar
>>>
```


10.1 OPERATORS FOR DICTIONARIES

- len (d) Find the number of key/value pairs in the dictionary
- del d[k] Delete a pairing from the dictionary
- k in d Returns true if the key is in the dictionary
- k not in d Returns true if the key is NOT in the dictionary
- d[k] = value Adds key/value pairing to the dictionary

```
# Find the number of key:value pairs in the dictionary
print(len(national_dish))

# Delete a pairing from the dictionary
del national_dish['Poland']

# Return true if a key is in the dictionary
print('Austria' in national_dish)

# Returns true if the key is NOT in the dictionary
print('Poland' not in national_dish)

# Adds a key;value pairing
national_dish['UK'] = 'Fish and Chips'
print('\n')

for k,v in national_dish.items():
    print('{0} :{1}'.format(k,v))
```

```
>>>
6
True
True


Austria :Weiner Schnitzel
Sweden :Kottbullar
Greece :Moussaka
Belgium :Moules-frites
UK :Fish and Chips
Germany :Currywurst
```

We can loop through the key/value pairings in a dictionary and print them out OR use them in a program in some other way.

Look at this simple test example:

This will loop through all the key/value pairs in the dictionary, checking the user input against the key.

```
for key, value in national_dish.items():
    print("Dish:" , value)
    country = input("Which country has this national dish? ")
    if key == country:
        print("Correct")
    else:
        print("Incorrect")
```


EXERCISE 26: DICTIONARIES

Write a program which uses at least two functions. The program should:

1. Create a dictionary with key/value pairings for revision of a subject of your choice, e.g. a glossary for computer science terms. Aim for at least six pairs in the dictionary.
2. The program should display the value and ask for the key, as in the example above.
3. The program must keep track of the correct answers and display the score, as a percentage, when all terms have been answered.
4. To extend this, your program should now keep asking for the definitions until the user has scored 60% or more.

10.2 NESTED DICTIONARIES

In the same way that we can nest lists inside another list, we can also do this with a dictionary, which will allow us to create data structures called 'records' which can be accessed using the key.

Example:

```
staff_cars = {'JKR':{'name':'Mr JK Rees','make':'Ford','model':'Focus','reg':'LN62 4FD'}}

# add to staff_cars dictionary
staff_cars['HJG']={'name':'Miss HJ Grove','make':'Vauxhall','model':'Astra','reg':'KN06 7LA'}

for ID,staff_info in staff_cars.items():
    print('\nStaff ID: {0}'.format(ID))
    for k in staff_info:
        print(k,': {0}'.format(staff_info[k]))
```

```
Staff ID: JKR
name : Mr JK Rees
reg : LN62 4FD
model : Focus
make : Ford

Staff ID: HJG
name : Miss HJ Grove
reg : KN06 7LA
model : Astra
make : Vauxhall
>>>
```

In this example, I have created a dictionary that will match staff car details to the staff ID codes.

The staff details are included as the value part of the key/value pairing and can be accessed by using the key for each record in the dictionary.

We can think about the keys inside the nested staff details dictionary as field names. We use field names when we create records in a database; this data can be accessed separately by using the field name to identify what data should be processed.

The example above uses a nested for loop to iterate through each key in the staff cars dictionary and then each key/value pairing in the dictionary nested inside the value part of each element.

```
for ID,staff_info in staff_cars.items():
    print('\nStaff ID: {0}'.format(ID))
    for k in staff_info:
        if k == 'name':
            print(k,': {0}'.format(staff_info[k]))
```

```
Staff ID: JKR
name : Mr JK Rees

Staff ID: HJG
name : Miss HJ Grove
```


The nested for loop to print all key/value pairings can be edited to access specific data in the nested dictionary by using the fieldname, i.e. the key of the item.

```
staff_id = input("Please enter the staff code: ").upper()
staff_name = input("Please enter staff name: ").title()
c_make = input("Please enter the make of car : ").title()
c_model = input("Please enter the model of car : ").title()
c_reg = input("Please enter the car registration : ").upper()

staff_cars[staff_id] = {'name':staff_name, 'make':c_make, 'model':c_model, 'reg':c_reg}

for ID, staff_info in staff_cars.items():
    print('\nStaff ID: {0}'.format(ID))
    for k in staff_info:
        print(k,': {0}'.format(staff_info[k]))
```

Adding more data to the dictionary can be achieved by simply replacing the value strings with variable names.

EXERCISE 27: NESTED DICTIONARY

Write a program that will use at least two functions. The program should:

1. Create a dictionary of student names and their homework scores out of 20.
2. Allow the teacher to add a student name and their homework marks.
3. Print out all records.

Test your program by entering the following sample data:

Name	**HW1**	**HW2**	**HW3**	**HW4**
James Farrow	15	12	18	11
Jenny Sullivan	8	12	13	16
Gemma Thompson	14	16	13	15
Oliver Booth	17	15	18	16
Kelly Jones	13	12	18	9

