

Dictionary

Quick Dictionary Exercise

Code

```
a = (("A",2), ("B",3), (1,4))  
dict_a = dict(a)  
print(dict_a)
```

```
b = [[1,"A"], [(2,3),4]]  
dict_b = dict(b)  
print(dict_b)
```

Output

```
{"A": 2, "B": 3, 1: 4}
```

```
{1: "A", (2, 3): 4}
```

Quick Dictionary Exercise

Code

```
a = (("A",2), ("B",3), (1,4))
dict_a = dict(a)
print(dict_a)
print(dict_a[2])
b = [[1,"A"], [(2,3),4]]
dict_b = dict(b)
print(dict_b)
print(dict_b[(2,3)])
```

Output

```
{"A": 2, "B": 3, 1: 4}
```

KeyError

```
{1: "A", (2, 3): 4}
```

```
4
```

Quick Dictionary Exercise

Code

```
for key in dict_b.keys():  
    print(key)  
  
for val in dict_b.values():  
    print(val)
```

Output

```
1  
(2, 3)  
  
A  
4
```

Quick Dictionary Exercise

Code

```
for k,v in dict_b.items():  
    print(k, v)
```

```
del dict_b[(2, 3)]  
print(dict_b)  
del dict_b[2]  
print(dict_b)
```

Output

```
1 A  
(2, 3) 4
```

```
{1: "A"}  
KeyError  
{1: "A"}
```

Quick Dictionary Exercise

Code

```
print(tuple(dict_a.keys()))  
print(list(dict_a.values()))  
  
dict_c = {1: {2: 3}, 4: 5}  
print(dict_c)
```

Output

```
("A", "B", 1)  
[2, 3, 4]  
  
{1: {2: 3}, 4: 5}
```

Quick Dictionary Exercise

Code

```
dict_d = dict_c.copy()  
dict_d[4] = 9  
dict_d[1][2] = 9  
print(dict_c)
```

```
del dict_c  
print(dict_c)
```

Output

```
{1: {2: 9}, 4: 5}
```

```
NameError
```

Dictionary

- Creation
 - Empty dictionary `{}`
 - Initialised dictionary `{key1: elem1, key2: elem2, ... }`
 - From tuple/list `dict(sequence_of_pairs)`
 - The element in tuple/list must be a pair
 - The first of the pair will be the key
 - The second of the pair will be the value
- Access
 - `dict_a[key]`
- Assignment
 - `dict_a[key] = value`

Dictionary

- Deletion
 - `del dict_a[key]` deletes the record corresponding to the specified key in the dictionary `dict_a`, if exists
 - `dict_a.clear()` removes all entries in dictionary `dict_a`
 - `del dict_a` deletes the dictionary

Dictionary

- Operations

- `dict_a.get(key, default=None)` returns `dict_a[key]` if exists or default value otherwise
- `key in dict_a` returns `True` if `key` in `dict_a`
- `dict_a.keys()` returns list of `dict_a` keys
- `dict_a.values()` returns list of `dict_a` values
- `dict_a.items()` returns list of `dict_a` (key,value)
- `len(dict_a)` returns number of elements

Dictionary Exercise

- Anagram
 - An anagram is a word or a phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once
 - Example:
 - “nag a ram” is an anagram for “anagram”
 - “eleven plus two” is an anagram for “twelve plus one”
 - Write a function `is_anagram(word1, word2)` that returns `True` if the two words are anagram of one another, otherwise returns `False`

By using
Dictionary

Dictionary Exercise

- T9
 - Old mobile phone only has numerical keypads where every letter is associated with a number as shown on the right (0 is space)
 - T9 is a predictive text technology for mobile phone
 - We can represent the keypad in two different ways
 - A list such that each element is a string of characters associated with the number which is the element's index which gives us [" ", "'", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"]
 - A dictionary where the keys are the characters and the values are the associated number which gives us {"a": 2, "b": 2, ..., "z": 9, " ", 0}



Dictionary Exercise

- T9
 - Suppose there are other alphabets and other symbols for larger numbers
 - Write a function `to_dict(keyL)` which take in the keys as the list representation and returns the dictionary representation
 - Write a function `to_list(keyD)` which take in the keys as the dictionary representation and returns the list representation



Dictionary Exercise

- T9
 - Write a function `to_nums(word)` that takes an input string and returns an integer representing the numbers to be pressed to input the string using T9
 - You may assume that both `keyL` and `keyD` are already initialized
 - Example: `to_nums("i luv u")` returns `4058808`
 - Write a function `to_letters(num)` that takes in a number and returns a list of all combinations of letters that can be represented by the numbers on a keypad in any order
 - You may assume that both `keyL` and `keyD` are already initialized

