**Python Dictionaries and Sets**

**Overview**

|  |
| --- |
| The data type dictionary fall under mapping. It is a mapping between a set of keys and a set of values. The key-value pair is called an item. A key is separated from its value by a colon(:) and consecutive items are separated by commas. Items in dictionaries are unordered, so we may not get back the data in the same order in which we had entered the data initially in the dictionary. Dictionaries are **mutable** which implies that the contents of the dictionary can be changed after it has been created. |

**Creating a dictionary**

To create a dictionary, the items entered are separated by commas and enclosed in curly braces. Each item is a key value pair, separated through colon (:). The keys in the dictionary must be unique and should be of any immutable data type, i.e., number, string or tuple.

**See three instances of creating dictionaries below**

*#creating an empty dictionary*marks= {}  
x=type(marks)  
print(x)  
*#using built-in function to create a dictionary*marks1= dict()  
print(type(marks1))  
*#mapping names of students to marks*marks3={**'Andy'**:63, **"Amy"**:95, **"Billy"**:58}  
print(type(marks3))

**Accessing items from a dictionary**

*#accessing values of the dictionary*print (marks3[**"Amy"**])

**Output**

95

Adding elements to the dictionary

*#Adding elements to the dictionary*marks3[**"Thapelo"**]= 82  
print(marks3)

**Output**

{'Andy': 63, 'Amy': 95, 'Billy': 58, 'Thapelo': 82}

**Modifying an Existing Item**

The existing dictionary can be modified by just overwriting the key-value pair. Example to modify a given item in the dictionary:

eg

Marks3[“Amy”]= 88

**Dictionary operations**

**Membership**

The membership operator in checks if the key is present in the dictionary and returns True, else it returns False.

y= **"Asaph" in** marks3  
print(y)

**Output**

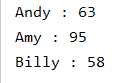
False

**Traversing a Dictionary**

Using a for loop to display key and values.

*#Traversing through a dictionary-Method 1***for** key **in** marks3:  
 print(key, **':'**, marks3[key])

Output



*# Traversing through a dictionary-Method 2***for** key, value **in** marks3.items():  
 print(key, **':'**, value)

**Dictionary methods and Built-in functions**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| len() | Returns the length or number of key: value pairs of the dictionary passed as the argument | print(len(marks3))  returns 3 |
| keys() | Returns a list of keys in the dictionary | print(marks3.keys())  **Output**  dict\_keys(['Andy', 'Amy', 'Billy', 'Ganief']) |
| values() | Returns a list of values in the dictionary | print(marks3.values())  dict\_values([63, 95, 58, 90]) |
| items() | Returns a list of tuples(key – value) pair | print( marks3.items())  dict\_items([('Andy', 63), ('Amy', 95), ('Billy', 58), ('Ganief', 90)]) |
| get() | Returns the value corresponding to the key passed as the argument If the key is not present in the dictionary it will return None | print( marks3.get(**"Amy"**))  **Output**  95 |
| update() | appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary | marks4={**"Mary"**:62, **"Margret"**:96} marks3.update(marks4) print(marks3.items())  Output  dict\_items([('Andy', 63), ('Amy', 95), ('Billy', 58), ('Ganief', 90), ('Mary', 62), ('Margret', 96)]) |

**Exercise**

**Task1**

Write a program to count the number of times a character appears in a given string.

**Task 2**

Write a function to convert a number entered by the user into its corresponding number in words. For example, if the input is 145 then the output should be ‘One Four Five.

**Sets in Python**

A Set is an unordered collection data type that is **iterable**, **mutable** and has **no duplicate** elements. Python’s set class represents the mathematical notion of a set. Since sets are unordered, we cannot access items using indexes like we do in lists.

myletters = set([**"a"**, **"b"**, **"c"**, **"d"**])  
print(myletters)

Output

{'d', 'b', 'c', 'a'}

If you run the above code the printout order might be different.

**Adding elements**

#creating a set

myletters = set([**"a"**, **"b"**, **"c"**, **"d"**])

#adding elements to a set  
myletters.add(**"k"**)  
print(**"Myletters:"**, end=**""**)  
print(myletters)

**Adding elements to a set using an iterator**

*#Adding elements to a set*myletters = set([**"a"**, **"b"**, **"c"**, **"d"**])  
**for** i **in** range(1,5):  
 myletters.add(i)  
print(**"Myletters:"**, end=**""**)  
print(myletters)

Output

Myletters:{1, 2, 3, 'a', 'd', 4, 'c', 'b'}

#### **Union**

Two sets can be merged using union() function or | operator.

*#Merging sets using union*myletters = set([**"a"**, **"b"**, **"c"**, **"d"**])  
mynumbers= set([8,9,10,11,12])  
result= myletters.union(mynumbers)  
print(**"Myletters:"**, end=**""**)  
print(result)

Note-Instead of keyword union, you can also use | character as shown below

result= myletters|mynumbers

#### **Intersection**

This can be done through intersection() or & operator.

*#Merging sets using intersection*myletters = set([**"a"**, **"b"**, **"c"**, **"d"**,8])  
mynumbers= set([8,9,10,11,12])  
result= myletters.intersection(mynumbers)  
print(**"Myletters:"**, end=**""**)  
print(result)

The alternative of intersection is & sign.

result= myletters&mynumbers

#### **Difference**

Difference() is used to check for the differences between sets. An alternative to difference() is the –sign as shown below

*#difference between sets using difference()*myletters = set([**"a"**, **"b"**, **"c"**, **"d"**,8])  
mynumbers= set([8,9,10,11,12])  
result= myletters.difference(mynumbers)  
print(**"Myletters:"**, end=**""**)  
print(result)

Output

Myletters:{'c', 'd', 'a', 'b'}

**Task1**

Declare a set to store five colours of your choice. Add Python program to add member(s) in a set. Use iterator to display the contents of the set.