MANIPAL ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

MDS5103/MSBA5104 Segment 04

TUPLES AND DICTIONARIES - TUTORIAL

Table of Contents

©COPYRIGHT 2022 (Ver. 1.0), ALL RIGHTS RESERVED. MANIPAL ACADEMY OF HIGHER EDUCATION

Introduction

Tuples and dictionaries are datatypes that store a collection of data. In this topic, we will look at the difference between tuples and dictionaries, the way they are used in Python and their other characteristics.

©COPYRIGHT 2022 (Ver. 1.0), ALL RIGHTS RESERVED. MANIPAL ACADEMY OF HIGHER EDUCATION

1. Tuples

Tuples are immutable objects. They are iterable, and hence they can be looped over. The elements of a tuple cannot be modified within the same object. This is its special characteristic. Like lists, tuples can contain heterogeneous elements. For example, a tuple can contain an integer as well as a string.

Unlike lists that are enclosed in a square bracket, tuples are enclosed in a round parenthesis as shown below.

```
# Example of a simple tuple
t=(1,2,3,5)
print(t)
```

Output

```
(1, 2, 3, 5)
```

In a tuple, elements can be accessed by indexing as shown below.

```
# To access the first element of a tuple, we can use print t[0] (square bracket).
print(t[0])
print (t[1])
```

Output

```
1
2
```

The main difference between a list and a tuple is that the lists are mutable, whereas the tuples are immutable. This means that the elements of a list can be modified while that of a tuple cannot. The code below demonstrates the same.

```
# To create a list we assign the elements using a square bracket.
li6=[1,2,3,4]
# In a list, The value at any position can be changed using an assignment operator.
li6[0]=0
# Any changes done to the individual element is reflected in the original list.
print (li6)
```

Output

```
[0 2 3 4]
```

Unlike in a list, the individual elements of a tuple cannot be changed. Attempting to change the elements of a tuple will result in an error as shown below.

```
t[0]=0
```

```
Type Error -----------------------------------------------
Traceback(most recent call last)

~\AppData\Local\Temp/ipykernel_2576/1315802387.py in <module>
1 # Unlike the list, individual elements of the tuple cannot be changed.
2 # Attempt to change elements of a tuple will result in an error.
----> 3 t[0]=0

TypeError: 'tuple' object does not support item assignment
```

The ‘dir()’ function can be used on the object to get a list of functions, methods, and attributes that can be used on them.

```
print (dir(t))
```

Output

```
        ['____add___', '____class', '___class_getitem_', '_contains_____', '______
_delattr____', '__dir__', '____doc_____', '__eq___', '_format__', '____ge_______', '
_____getattribute', '_getitem_', '___getnewargs___', '__gt___', '____hash____', '
_init_',
'______________init_subclass______', '__iter___', '___le____', '__len___', '___lt__', '
______________mul__',
'____ne____', '___new__', '______reduce__________', '____reduce_ex__', '____repr_____', '
____rmul__',
'________setattr_______','____sizeof_____','____________str___','_____subclasshook
_________', 'count',  'index']
```

2. Dictionaries

Dictionaries are key-value pairs, and are created using curly brackets. The keys themselves are immutable objects. However, the values within the keys can be updated, selected, or deleted using the keys. Unlike lists, which are indexed using a range of numbers, dictionaries are indexed using unique keys.

Consider the example in which we maintain the names and incomes as two lists. We must index the lists each time to correlate the name and the person. For example, to calculate the income of Ramesh, identify the index of 'Ramesh' and use it to index the income list.

```
Income = [30000,40000,25000]
name = ["Umesh", "Ramesh","Nilesh"]
in_r = name.index('Ramesh')
Income[in_r]
```

Output

```
40000
```

This can be done more efficiently by using dictionaries. Dictionaries represent data as a key-value pair. The name of a person can be used as a key and the income as a value. The code to demonstrate the same is shown below. Here, "Umesh" is the key and 30000 is the value.

```
Inc = {"Umesh":30000,"Ramesh":40000,"Nilesh":25000}
print(Inc)
```

Output

```
{'Umesh': 30000, 'Ramesh': 40000, 'Nilesh': 25000}
```

The values in the dictionary can be accessed using the format: dict_name[key].

```
print(Inc["Umesh"])
```

Output

```
30000
```

The 'keys()' method displays the list of keys in the dictionary. The 'values()' method displays the list of values in the dictionary.

```
print(Inc.keys())
print(Inc.values())
```

Output

```
dict_keys(['Umesh', 'Ramesh', 'Nilesh'])

dict_values([30000, 40000, 25000])
```

To check if a key is present in the dictionary, we can use the 'in' keyword as shown below.

```
"Nilesh" in Inc
```

Output

```
True
```

Since “Nilesh” is a key in the dictionary, the above output is True.

Values corresponding to the keys can be changed using the assignment operator as shown below. The output will have the new value assigned.

```
Inc["Nilesh"] = 35000
print(Inc)
```

Output

```
{'Umesh': 30000, 'Ramesh': 40000, 'Nilesh': 35000}
```

The ‘del()’ function can be used to delete a specific key. Note that the deletion of a key will result in the deletion of the value as well.

```
del(Inc["Nilesh"])
```

```
print(Inc)
```

Output

```
{'Umesh': 30000, 'Ramesh': 40000}
```

The ‘dir()’ function on the dictionary will provide a list of methods, attributes and functions available to manipulate the dictionary object.

```
print (dir(Inc))
```

Output

```
['    class ', '   class_getitem    ', '    contains ', '   delattr      ', '
   delitem ',  ' dir ', '   doc     ', '      eq    ', '  format   ', '      ge    ', '
   getattribute    ',

' getitem ', 'gt    ', '   hash  ', 'init    ', '    init_subclass ', '    ior     ', '          iter
   ', '    le    ', 'len  ', '       lt   ', 'ne     ', '    new    ', '  or  ', '    reduce ', '
   reduce_ex', '   repr', '  reversed   ', '        ror', '  setattr    ', '      setitem   ', '
   sizeof    ', 'str   ', '   subclasshook ', 'clear', 'copy', 'fromkeys', 'get', 'items',
'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

To access the key/value pairs, use the ' items()' method.

```
print (Inc.items())
```

Output

```
dict_items([('Umesh', 30000), ('Ramesh', 40000)])
```

Values of a key can be another dictionary. It is possible to have a 'Dictionary of Dictionaries'. A sample code to demonstrate the same is shown below. The key value for 'Income' for the key 'Ramesh' can be obtained by multi-level slicing. Similarly, a new dictionary { 'Gender':'female', 'Income':30000 } can be assigned to a new key "Seema" as shown below.

```
Income1 = { 'Ramesh': { 'Gender':'male', 'Income':40000},
            'Ana': { 'Gender':'female', 'Income':66000 },
            'Uma': { 'Gender':'female', 'Income':20000 },
            'Umesh': { 'Gender':'male', 'Income':58000 } }
# Print the Income of  Ramesh
print(Income1['Ramesh']['Income'])

#  To create a new entry,  First -  create sub-dictionary data
income2 = { 'Gender':'female',  'Income':30000 }

# Add  data  to  original  dictionary  under  key  'Seema'
Income1['Seema'] = income2
print(Income1)
```

Output

```
        40000

         {'Ramesh': {'Gender': 'male', 'Income': 40000}, 'Ana': {'Gender': 'female',

         'Income': 66000}, 'Uma': {'Gender': 'female', 'Income': 20000}, 'Umesh':

        {'Gender': 'male', 'Income': 58000}, 'Seema': {'Gender': 'female', 'Income':

   30000}}
```

Dictionary is iterable. Hence, we can use a ‘for’ loop on a dictionary as shown in the code below. It displays the keys of the dictionary.

```
    for  x  in  Inc:
         print (x)
```

Output

```
        Umesh Ramesh
```

As shown in the output above, Iteration returns the key. Therefore, we can access the values using the dict_name[key]. The code below demonstrates the same.

```
for x in Inc:
    print (x, Inc[x])
```

Output

```
Umesh 30000

Ramesh 40000
```

Similar to list comprehension, we can create new dictionaries using **curly braces** and a ' for' loop. For example, we can create a key/value pair of a positive and negative of a number using comprehensions as shown below.

```
pos_neg = {I:-I for I in range(3)}
print(pos_neg)
type(pos_neg)
```

Output

```
{0:0, 1:-1, 2:-2}

dict
```

3. Case Study - Frequency Count

In this case study, we are going to find the occurrence of each word in a text string. Create a list of each word in the string, and loopover this list to create a dictionary of words and their occurrences.

```
assign_txt="Narendra Damodardas Modi, born 17 September 1950) is the 15th and \
 current Prime Minister of India, in office since 26 May 2014. Modi, a leader \
  of the Bharatiya Janata Party was the Chief Minister of Gujarat from 2001 to \
  2014 and is the Member of Parliament from Varanasi. He led the BJP in the \
  2014 general election, which gave the party a majority in the Lok Sabha, the \
  first for any political party in India since 1984. As the Chief Minister of \
  Gujarat, Modi's economic policies were praised, while his administration was \
  also criticised for failing to significantly improve the human development \
  in the state, and for failing to prevent the 2002 Gujarat riots. A Hindu \
  nationalist and member of the Rashtriya Swayamsevak Sangh, Modi, remains a \
  controversial figure domestically and internationally. Modi was born on 17 \
  September 1950, to a family of grocers in Vadnagar, Mehsana district, Bombay \
  State (present-day Gujarat).Modi's family belonged to the Modh-Ghanchi-Teli \
  (oil-presser) community, which is categorised as an Other Backward Class by \
  the Indian government. Modi was the third of six children born to Damodardas \
  Mulchand (1915–1989) and Heeraben Modi (b. c. 1920). As a child, Modi helped \
  his father sell tea at the Vadnagar railway station, and later ran a tea \
  stall with his brother near a bus terminus. Modi completed his higher \
  secondary education in Vadnagar in 1967, where a teacher described him as an \
  average student and a keen debater, with an interest in theatre. Modi had an \
  early gift for rhetoric in debates, and this was noted by his teachers and \
  students. Modi preferred playing larger-than-life characters in theatrical \
  productions, which has influenced his political image. At age eight, Modi \
  discovered the Rashtriya Swayamsevak Sangh (RSS), and began attending its \
  local shakhas (training sessions). There, Modi met Lakshmanrao Inamdar, \
  popularly known as Vakil Saheb, who inducted him as an RSS balswayamsevak \
  (junior cadet) and became his political mentor. While Modi was training with \
  the RSS, he also met Vasant Gajendragadkar and Nathalal Jaghda, Bharatiya \
  Jana Sangh leaders who were founding members of the BJP's Gujarat unit in \
  1980. Engaged while still a child to a local girl, Jashodaben Narendrabhai \
  Modi, Modi rejected the arranged marriage at the same time he graduated from \
  high school. The resulting familial tensions contributed to his decision to \
  leave home in 1967. Modi spent the ensuing two years travelling across \
  Northern and North-eastern India, though few details of where he went have \
  emerged. In interviews, Modi has described visiting Hindu ashrams founded by \
  Swami Vivekananda: the Belur Math near Kolkata, followed by the Advaita \
  Ashrama in Almora and the Ramakrishna mission in Rajkot. Modi remained only \
  a short time at each, since he lacked the required college education."
```

The code below demonstrates the same. We use the split(“ “) function and use the word as the key of the dictionary and the count as the value. Every time a word is encountered, if it is part of the keys, the value is incremented by 1. If the word is not part of the keys, the value of the key is initialised to 1.

```
# Initialize the dictionary
d={}

# To get each word in the text , use the method split with delimiter as " ".
for w in assign_txt.split(" "):
    # if the word is present as a key in the dictionary, increase the value by 1
    if w in d:
        d[w]=d[w]+1
    # if word is not present as a key, add the key and set the value to 1
    else:
        d[w]=1
```

```
print (d)
```

The sample output of the above code is shown below. All the words form the key to the dictionary and the corresponding values are the number of occurrences.

Output

```
{'Narendra': 1, '': 262, 'Damodardas': 2, 'Modi,': 2, 'born': 3, '17': 2, 'September': 2, '1950)':
1, 'is': 3, 'the': 28, '15th': 1, 'and': 15, 'current': 1, 'Prime': 1, 'Minister': 3, 'of': 10, 'India,': 2,
'in': 15, 'office': 1, 'since': 3, '26': 1, 'May': 1, '2014.odi,': 1, 'a': 12, 'leader': 1, 'Bharatiya': 2,
'Janata': 1, 'Party': 1, 'was': 6, 'Chief': 2, 'Gujarat': 3, 'from': 3, '2001': 1, 'to': 9, '2014': 2,
'Member': 1, 'Parliament': 1, 'Varanasi.': 1, 'He': 1, 'led': 1, 'BJP': 1, 'general': 1, 'election,': 1,
'which': 2, 'gave': 1, 'party': 2, 'majority': 1, 'Lok': 1, 'Sabha,': 1, 'first': 1, 'for': 4, 'any': 1,
'political': 3, 'India': 1, '1984.As': 1, 'Gujarat,': 1, "Modi's": 1, 'economic': 1, 'policies': 1,
'were': 2, 'praised,': 1, 'while': 2, 'his': 8, 'administration': 1, 'also': 2, 'criticised': 1, 'failing':
2, 'significantly': 1, 'improve': 1, 'human': 1, 'development': 1, 'state,': 1, 'prevent': 1, '2002':
1, 'riots.': 1, 'A': 1, 'Hindu': 2, 'nationalist': 1, 'member': 1, 'Rashtriya': 2, 'Swayamsevak': 2,
'Sangh,': 1, 'Modi,remains': 1, 'controversial': 1, 'figure': 1, 'domestically': 1,
'internationally.': 1, 'Modi': 9, 'on': 1, '1950,': 1, 'family': 2, 'grocers': 1, 'Vadnagar,': 1,
'Mehsana': 1, 'district,': 1, 'Bombay': 1, 'State': 1, '(present-day': 1, "Gujarat).Modi's": 1,
'belonged': 1, 'Modh-Ghanchi-Teli': 1, '(oil-presser)': 1, 'community,hich': 1, 'categorised':
1, 'as': 4, 'an': 5, 'Other': 1, 'Backward': 1, 'Class': 1, 'by': 4, 'Indian': 1, 'government.Modi': 1,
'third': 1, 'six': 1, 'children': 1, 'Mulchand': 1, '(1915–1989)': 1, 'Heeraben': 1, '(b.': 1, 'c.': 1,
'1920).As': 1, 'child,': 1, 'helped': 1, 'father': 1, 'sell': 1, 'tea': 2, 'at': 3, 'Vadnagar': 2, 'railway':
1, 'station,': 1, 'later': 1, 'ran': 1, 'stall': 1, 'with': 3, 'brother': 1, 'near': 2, 'bus': 1,
'terminus.Modi': 1, 'completed': 1, 'higher': 1, 'secondary': 1, 'education': 1, '1967,': 1,
'where': 2, 'teacher': 1, 'described': 2, 'him': 2, 'average': 1, 'student': 1, 'keen': 1, 'debater,':
1, 'interest': 1, 'theatre.Modi': 1, 'had': 1, 'early': 1, 'gift': 1, 'rhetoric': 1, 'debates,': 1, 'this': 1,
'noted': 1, 'teachers': 1, 'students.Modi': 1, 'preferred': 1, 'playing': 1, 'larger-than-life': 1,
'characters': 1, 'theatrical': 1, 'productions,': 1, 'has': 2, 'influenced': 1, 'image.At': 1, 'age':
1, 'eight,': 1, 'discovered': 1, 'Sangh': 2, '(RSS),': 1, 'began': 1, 'attending': 1, 'its': 1, 'local': 2,
'shakhas': 1, '(training': 1, 'sessions).': 1, 'There,': 1, 'met': 2, 'Lakshmanrao': 1, 'Inamdar,':
1, 'popularly': 1, 'known': 1, 'Vakil': 1, 'Saheb,': 1, 'who': 2, 'inducted': 1, 'RSS': 1,
'balswayamsevak': 1, '(junior': 1, 'cadet)': 1, 'became': 1, 'mentor.While': 1, 'training': 1,
'RSS,': 1, 'he': 4, 'Vasant': 1, 'Gajendragadkar': 1, 'Nathalal': 1, 'Jaghda,': 1, 'Jana': 1,
'leaders': 1, 'founding': 1, 'members': 1, "BJP's": 1, 'unit': 1, '1980.Engaged': 1, 'still': 1,
'child': 1, 'girl,': 1, 'Jashodaben': 1, 'Narendrabhai': 1, 'rejected': 1, 'arranged': 1, 'marriage':
1, 'same': 1, 'time': 2, 'graduated': 1, 'high': 1, 'school.The': 1, 'resulting': 1, 'familial': 1,
'tensions': 1, 'contributed': 1, 'decision': 1, 'leave': 1, 'home': 1, '1967.Modi': 1, 'spent': 1,
'ensuing': 1, 'two': 1, 'years': 1, 'travelling': 1, 'across': 1, 'Northern': 1, 'North-eastern': 1,
'though': 1, 'few': 1, 'details': 1, 'went': 1, 'have': 1, 'emerged.In': 1, 'interviews,': 1, 'visiting':
1, 'ashrams': 1, 'founded': 1, 'Swami': 1, 'Vivekananda:': 1, 'Belur': 1, 'Math': 1, 'Kolkata,': 1,
'followed': 1, 'Advaita': 1, 'Ashrama': 1, 'Almora': 1, 'Ramakrishna': 1, 'mission': 1, 'Rajkot.':
1, 'remained': 1, 'only': 1, 'short': 1, 'each,': 1, 'lacked': 1, 'required': 1, 'college': 1,
'education.': 1}
```

E-references:

- Dive into Python by Mark Pilgrim

  Link: https://diveintopython3.net/

- Automate the Boring Stuff with Python by Al Sweigart

  Link: https://automatetheboringstuff.com/#toc

©COPYRIGHT 2022 (Ver. 1.0), ALL RIGHTS RESERVED. MANIPAL ACADEMY OF HIGHER EDUCATION