

Tuples

Chapter 12

Today's Outline

- Dictionary Exercise!
- Let's talk tuples.
- Let's talk about accessing tuple elements.
- Hey, what do you know, a zip function!
- for loops & tuples

Dictionary Methods



- What's the output for...

- **keys()**

- `inventory.keys()` # In shell
- `print(inventory.keys())` # In script window
- Returns a list! Notice the square brackets []

- **values()**

- `inventory.values()` # In shell
- `print(inventory.values())` # In script window
- Returns a list! Notice the square brackets []

Dictionary Methods



- What's the output for...

- `items()`

- `inventory.items()` # In shell
- `print(inventory.items())` # In script window
- Returns a tuple! (We'll talk about that next class)

- `clear()` –

- `dict.clear(inventory)`

Traversing a Dictionary



- Create this dictionary for a map legend.

```
legend = {0: 'no value', 1: 'deciduous', 2: 'conifers',  
3: 'industrial', 4: 'residential', 5: 'water bodies',  
6: 'agricultural' }
```

- Write a function that will print all the items using the following structure

```
Define a function that takes a dictionary name as an argument:  
for every (key, value) in dictionary - hint: use .items():  
    print the key and value
```

Traversing a Dictionary



- Here's my answer

```
def printDictionary(dicInput):  
    for (key, value) in dicInput.items():  
        print key, value
```

```
printDictionary(legend)
```

Traversing a Dictionary



Write a function that will print all the keys... using the same structure:

```
def printKeys(dicInput) :  
    for key in dicInput.keys() :  
        print(key)
```

```
printKeys(legend)
```

Traversing a Dictionary



Write a function that will print all the values... using the same structure:

```
def printValues(dicInput):  
    for value in dicInput.values():  
        print(value)
```

```
printValues(legend)
```




Dictionary Exercise

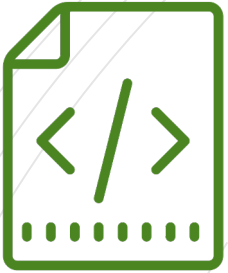
- Create a histogram (shows the frequency of data) to show how many letters appear in a word.
- The output should be a dictionary. The method `dict()` creates an empty dictionary.
- So if you make a variable like `d = dict()` then anything you put into `d` will be output in a dictionary.
- Remember how to add to a dictionary:

```
d['key'] = 'value'
```



Dictionary Exercise

- Use the following structure:
 - # Define function called histogram that takes string for an argument
 - # create an empty dictionary.
 - # for a letter in a word:
 - # if that letter is not in the empty dictionary:
 - # create a key/value in that dictionary.
 - # or else:
 - # if that letter *is* in the dictionary, increment value of the key/value pair by 1. (hint += 1)
 - # return the final dictionary



Exercise Answer

- My solution

```
def histogram(word) :  
    d = dict()  
    for letter in word:  
        if letter not in d:  
            d[letter] = 1  
        else:  
            d[letter] += 1  
    return d
```

```
print(histogram('Python'))
```

Compound Data Types

- **Strings** `'Hello World'`
 - made up of small pieces - characters
- **Lists** `['Hello', 'World']`
 - made up of small pieces - elements
- **Dictionaries** `{'word1': 'Hello', 'word2': 'Hello'}`
 - Made up of key / value pairs.
- **Tuples** `('Hello', 'World')`
 - Similar to lists: made up of small pieces - elements

Tuples...

- A tuple is similar to a list except that it is **immutable**.
- This means that methods like `sort` and `reverse` won't work on a tuple!
- Lists are more common than tuples. But there are some cases in which you will want a tuple:
 - Primary reason is that you don't want the dataset to change!
- **Tuple** is defined within parentheses and the syntax is:
 - `Tuple = ('a', 'b', 'c', 'd', 'e')`

Tuples...

- Tuple with a single element must include the final comma.
 - `tuple = ('a',)`
 - `type(tuple)` #output is a tuple!
- Without the comma, Python treats 'a' as a string
 - `tuple2 = ('a')`
 - `type(tuple2)` #output is a string!

Accessing Elements



- Similar to accessing List elements. Use the index.
 - `mytuple = ('a', 'b', 'c', 'd', 'e')`
- How would I get just 'a'?
 - `mytuple[0]`
- How would I get 'b' and 'c'?
 - `mytuple[1:3]`

zip Function

- You can use zip to combine strings and lists into tuples. Try this in shell:

```
string = 'abc'  
list = [0, 1, 2]  
zip(string, list)
```
- What's the output?

```
[('a', 0), ('b', 1), ('c', 2)]
```
- Shorter list will define length if they are unequal.

for loops & Tuples



- Traverse a list of tuples
- Write a script with a for loop to print each letter and number

```
mytuple = [('a',0), ('b',1), ('c',2)]
```

```
for letter, number in mytuple:  
    print number, letter
```

Summary

- Creating new dictionaries with `dict()`!
- Tuples are similar to lists but are **immutable**.
- Accessing tuple elements similar to lists.
- Zip function is a way to create tuples
- Iterate through tuples using a for loop.