

Python

Python and R for Data Science

Data Science and Management



Exercise 1: find number of unique characters

Define a function `count_uniq` that:

- takes as arguments:
 - a string `s`
- returns:
 - the number of unique characters in `s`

In [6]: *# Solution goes here*

Test your code

Run this code to test your solution:

```
In [7]: try: assert count_uniq("test") == 3 and count_uniq("Aejeje") == 3 and r
except: print('Test failed')
```

Test failed

Exercise 2: remove duplicates

Define a function `remove_duplicates` that:

- takes as arguments:
 - a list `s` of strings
- returns:
 - a copy of `s` without duplicate elements

In [6]: *# Solution goes here*

Test your code

Run this code to test your solution:

```
In [7]: try: assert remove_duplicates(["test", "luiss", "data", "test", "science"])  
except: print('Test failed')
```

Test failed

Exercise 3: find common elements

Define a function `common_elements` that:

- takes as arguments:
 - a set `s` of strings
 - a set `k` of strings
- returns:
 - a list containing all common elements between `s` and `k`

In [8]: *# Solution goes here*

Test your code

Run this code to test your solution:

```
In [9]: friend1_companies = {'Google', 'Amazon', 'Apple', 'Microsoft'}  
friend2_companies = {'Facebook', 'Google', 'Tesla', 'Amazon'}  
try: assert common_elements(friend1_companies, friend2_companies) == {'  
except: print('Test failed')
```

Test passed

Exercise 4: count word frequency

Define a function `word_freq` that:

- takes as arguments:
 - a string `s`
- returns:
 - a dictionary containing as key each word in `s` and as value the count of that word in `s`

Count the word case-insensitive.

```
In [12]: # Solution goes here
```


Test your code

Run this code to test your solution:

```
In [13]: try: assert word_freq("Python is fun and learning Python is fun") == {'  
except: print('Test failed')
```

Test failed

Exercise 5: track voting results

Define a function `update_votes` that:

- takes as arguments:
 - a dictionary `votes` having as key names of candidates and as value the number of votes received by each one of them
 - a list `new_votes` of names of candidates
- returns:
 - the `votes` dictionary updated with the new votes received

```
In [15]: # Solution goes here
```

Test your code

Run this code to test your solution:

```
In [16]: votes = {
    'Alice': 120,
    'Bob': 150,
    'Charlie': 90
}

new_votes = ['Alice', 'Charlie', 'Charlie', 'Bob', 'Alice', 'Alice']
try: assert update_votes(votes, new_votes) == {'Alice': 123, 'Bob': 151}
except: print('Test failed')
```

Test failed

