

Introduction to Python



Sample Exam Questions

Note: Star rating on top right-hand corner denotes question difficulty

String Exercises

Try it!



You are looking to delete a certain character at a specific index of a string. Write a function, `remove_character`, that takes in two arguments, a string and an integer `n`, and deletes the `n`-th index character from that string.

For example, `remove_character('gone', 2)` returns `'goe'` and `remove_character('disappear', 4)` returns `'disapear'`.

Solution



```
def remove_character(word, index):  
    return word[:index] + word[index+1:]
```

Try it!



Let's make it a little more challenging. Now, remove the character at every n-th index of the given string. We still want to

For example, `remove_character('vanish', 2)` returns `'vaih'` **and** `remove_character('disappear', 3)` returns `'disppar'`.

Solution



```
def remove_character(word, index):  
    new_string = word[0]  
    for i in range(len(word)):  
        if i % index != 0:  
            new_string += word[i]  
    return new_string
```


Try it!



We are only interested in the first three characters of each word. Write a Python function, `first_three`, that keeps only the first three characters. If the given string is less than three characters long, return the original string.

For example, `first_three('Disney')` returns `'Dis'` and `first_three('OK')` returns `'OK'`.

Solution



```
def first_three(word) :  
    if len(word) > 3:  
        return word[:3]  
    else:  
        return word
```


Try it!



You are given a string containing several words separated by colons, however, you decide that a list of words would be more helpful in your further analysis. Write a function, `string_converter`, that converts a long string into a list of words.

Solution



```
def string_converter(sentence):  
    return sentence.split(':')
```


Try it!



Write a Python function, `verb`, that adds an “-ing” to the end of the input string. However, if the string already ends with an “ing”, then add “ly” to the end of that word instead.

For example, `verb('accept')` returns `'accepting'`,
`verb('do')` returns `'doing'` and
`verb('accepting')` returns `'acceptingly'`.


Incorrect Solution



```
def verb(word):  
    if len(word) > 2:  
        if word[-3:] == 'ing':  
            word += 'ly'  
        else:  
            word += 'ing'  
    return word
```


Why is this incorrect?

Solution



```
def verb(word):  
    if len(word) > 2:  
        if word[-3:] == 'ing':  
            word += 'ly'  
    else:  
        word += 'ing'  
    return word
```

Solution



```
def verb(word) :  
    is_verb = False  
    if len(word) > 2:  
        if word[-3:] == 'ing':  
            is_verb = True  
    if is_verb:  
        return word + 'ly'  
    else:  
        return word + 'ing'
```

Try it!



You are interested in finding out what the first non-repeating character is in a word. Write a function, `no_repeat`, that returns the first character that satisfies the criteria. Return `None` if no character fulfills this condition.


For example, `no_repeat('anteater')` returns `'n'` while `no_repeat('aardvark')` returns `'d'`.

Hint: There are multiple ways to approach this problem.

Solution


```
def no_repeat(word):
    char_order = []
    counter = {}
    for c in word:
        if c in counter:
            counter[c] += 1
        else:
            counter[c] = 1
            char_order.append(c)
    for c in char_order:
        if counter[c] == 1:
            return c
    return None
```


Solution



```
def no_repeat(word):  
    for i in range(len(word)):  
        char = word[i]  
        rest_of_word = word[:i] + word[i+1:]  
        if char in rest_of_word:  
            continue  
        else:  
            return char
```

Solution



```
def no_repeat(word):  
    for char in word:  
        count = word.count(char)  
        if count == 1:  
            return char
```

We didn't cover this: Let's google it together!
A string method that returns the number of elements with the specified value (works for lists too!)

Try it!



A common typography mistake is to have duplicate words in a sentence. Write a function, `remove_duplicates`, that takes a sentence as input and returns the input sentence but removing words that occur more than once.

For example, `remove_duplicates('This knocks my socks socks off')` **returns** `'This knocks my socks off'`.

Solution



```
def remove_duplicates(sentence):  
    list_of_words = sentence.split()  
    words = []  
    for word in list_of_words:  
        if word not in words:  
            words.append(word)  
    return ' '.join(words)
```


Try it!



Have you accidentally forgot that your Caps lock was already turned on and you typed your whole sentence in the opposite case? Use your Python knowledge to write a function, `swap_case`, that reverses the case for all characters in a given input string.

For example, `swap_case('tHIS IS ANNOYING.')` returns `'This is annoying.'`

Solution



```
def swap_case(sentence):  
    result_str = ""  
    for char in sentence:  
        if char.isupper():  
            result_str += char.lower()  
        else:  
            result_str += char.upper()  
    return result_str
```

Try it!



You want to determine what the longest and shortest word is in a particular sentence you have written but you discover that your word processor does not provide this information for you. Write a function, `shortest_longest_word`, that takes in a sentence and returns the following information: longest word, length of longest word, shortest word and length of shortest word. If there are words with equal length, choose the word that appears first in the sentence.

For example, `shortest_longest_word('The quick brown fox jumps over the lazy dog')` **returns** `('quick', 5, 'The', 3)` .

Solution

```
def shortest_longest_word(sentence):
    max_length = 0
    min_length = 20
    longest_word = ""
    shortest_word = ""
    for word in sentence.split():
        if len(word) > max_length:
            max_length = len(word)
            longest_word = word
        if len(word) < min_length:
            min_length = len(word)
            shortest_word = word
    return longest_word, max_length, shortest_word, min_length
```


Loops & If/Else Exercises

Try it!



Write a Python function, `factorial`, that takes an integer and return its factorial. A factorial is the product of an integer and all the integers less than it. You can assume that only a positive number will be passed in as an argument.

So, $5!$ (five factorial) = $5*4*3*2*1 = 120$. You can verify that your code returns the same result.

Solution



```
def factorial(num):
```

```
    factorial = 1
```

```
    for i in range(1, num + 1):
```

```
        factorial *= i
```

```
    return factorial
```

Recall: This means

`factorial = factorial * i`


Try it!



From a list that contains elements of any data types, write a function, `sorter`, that returns three lists that contain all the integer, string and float elements respectively.

For example, `sorter([1, 'a', 1.0, None, True, 2.2, 4])` returns `([1, 4], ['a'], [1.0, 2.2])`.

Solution



```
def sorter(list_):
    integer = list()
    string = list()
    flt = list()
    for elem in list_:
        if type(elem) == int:
            integer.append(elem)
        elif type(elem) == str:
            string.append(elem)
        elif type(elem) == float:
            flt.append(elem)
        else:
            continue
    return integer, string, flt
```

Try it!




Write a function, `common_elements`, that takes an input two lists and returns `True` if the lists share at least one common element and `False` otherwise.

For example, `common_elements([1, 2, 3, 4, 5], [5, 6, 7])` returns `True` while `common_elements(['e', 'e', 'l'], ['a', 'n', 't'])` returns `False`.


Solution

Note: This solution is inefficient as it continues the for loop even when we've found that $x == y$



```
def common_elements(list1, list2):  
    result = False  
    for x in list1:  
        for y in list2:  
            if x == y:  
                result = True  
    return result
```

(Better) Solution



```
def common_elements(list1, list2):  
    for x in list1:  
        for y in list2:  
            if x == y:  
                return True  
    return False
```



Try it!



Now, can you instead return a list of the common elements from the two input lists?

For example, `common_elements([1, 2, 3, 4, 5], [5, 6, 7])`
returns `[5]` while
`common_elements(['e', 'e', 'l'], ['a', 'n', 't'])`
returns `[]`.

Solution



```
def common_elements(list1, list2):  
    result = []  
    for x in list1:  
        for y in list2:  
            if x == y:  
                result.append(x)  
    return result
```

Try it!



Your international colleagues are new to this country and are not accustomed to the imperial system of unit measurements. Write a function, `converter`, that can convert degrees in Celsius to Fahrenheit and vice versa.

The input to this function is a string with two components: temperature (in float) followed by the unit (either C or F). Depending on the input, perform the appropriate conversion to the other unit:


$$F = (9/5 * C) + 32$$

$$C = 5/9 * (F - 32)$$

Try it!

For example, `converter('100C')` returns `212.0` and
`converter('100F')` returns `37.778`.

Solution



```
def converter(temp):  
    degree = float(temp[:-1])  
    if temp[-1].upper() == 'C':  
        result = (9 * degree) / 5 + 32  
    elif temp[-1].upper() == 'F':  
        result = (degree - 32) * 5 / 9  
    return result
```


Try it!



You are interested in finding out the number of digits and letters in a particular string. Write a function, `counter`, that takes in an input string and returns the count of digits, followed by letters.

For example, `counter('Cool function')` returns `(0, 12)` while `counter('I ate 13 bagels')` returns `(2, 10)`.

Solution



```
def counter(sentence):  
    digits = 0  
    letters = 0  
    for char in sentence:  
        if char.isdigit():  
            digits += 1  
        elif char.isalpha():  
            letters += 1  
    return digits, letters
```


Try it!



You are building a program that determines the median from three numerical inputs (float/integer).

For example, `median(20, 30, 40)` returns 30 while `median(15, 15, 200)` returns 15.

Solution



```
def median(a,b,c):  
    if a > b:  
        if a < c:  
            median = a  
        elif b > c:  
            median = b  
        else:  
            median = c  
    else:  
        if a > c:  
            median = a  
        elif b < c:  
            median = b  
        else:  
            median = c  
    return median
```

many ways to order the
if/else statements!


Try it!- Let's go further



You are building a program that determines the median from a list that can contain any number of numerical elements.

For example, `median([20, 30, 40])` returns 30 while `median([15, 15, 15, 200, 30, 20])` returns 17.5.

Solution



```
def median(list_):  
    list_.sort()  
    l = len(list_)  
    mid = (l-1) // 2  
  
    if (l%2 == 0):  
        return (list_[mid] + list_[mid+1]) / 2  
    else:  
        return list_[mid]
```


Try it!



Write a Python function, `digit_sum`, that takes as input a whole number and returns the sum of the individual digits of that number.

For example, `digit_sum(888)` returns 24 while `digit_sum(100)` returns 1.

Solution



```
def digit_sum(num) :  
    number = str(num)  
    count = 0  
    for digit in number:  
        add = int(digit)  
        count += add  
    return count
```

Try it!



You are provided with two separate strings. Write a function, `anagram`, that returns `True` if the second string is an anagram of the first string and `False` otherwise. You will need to make your function case-insensitive. Note that the number of whitespaces must also be the same for two strings to be anagrams of each other.

For example, `anagram('listen', 'silent')`,
`anagram('Edward Daniels', 'andrew Laeddis')` and
`anagram('Rachel solando', 'Dolores chanal')` all
returns `True`.

Solution



```
def anagram(str1, str2):  
    str1_list = list(str1.lower())  
    str2_list = list(str2.lower())  
    str1_list.sort()  
    str2_list.sort()  
    return (str1_list == str2_list)
```

List Exercises

Try it!



Write a Python function, `reverse_list`, that takes a list and reverses its order.

For example, `reverse_list([1,2,3])` returns `[3,2,1]`
while `reverse_list(['c','i','v','i','c'])` returns
`['c','i','v','i','c']`

Solution



```
def reverse_list(list_):  
    reverse = []  
    for i in range(len(list_)):  
        reverse.append(list_[len(list_)-i-1])  
    return reverse
```

Solution



```
def reverse_list(list_):  
    return list_[::-1]
```

slicing operator: -1 means we decrease the index by 1 each time

Solution



```
def reverse_list(list_):  
    list_.reverse()  
    return list_
```

changes the existing list by
reversing its elements

Dictionary Exercises

Try it!



Scrabble is a word game where points are earned through forming words. The points from a word is obtained by adding the points from each individual letter. Write a function, `scrabble_score`, that takes a word as input and returns the scrabble score for that word. Your function should be case insensitive.

For example, `scrabble_score('Hello')` returns 8 and `scrabble_score('Scrabble')` returns 14.

The dictionary on the next slide gives the points for each letter.

Try it!


```
score = {"a": 1, "c": 3, "b": 3, "e": 1, "d": 2,  
"g": 2, "f": 4, "i": 1, "h": 4, "k": 5, "j": 8,  
"m": 3, "l": 1, "o": 1, "n": 1, "q": 10, "p": 3,  
"s": 1, "r": 1, "u": 1, "t": 1, "w": 4, "v": 4,  
"y": 4, "x": 8, "z": 10}
```

Solution



```
def scrabble_score(word) :  
    word = word.lower()  
    total = 0  
    for letter in word:  
        total += score.get(letter)  
    return total
```


(Safer) Solution



```
def scrabble_score(word) :  
    word = word.lower()  
    total = 0  
    for letter in word:  
        total += score.get(letter, 0)  
    return total
```

an optional argument that specifies a value to return if the key does not exist

Try it!



You are managing your company's database of customer information. The information is stored as dictionary and a sample entry is shown below.

```
customer_info = {  
    'name': 'Kelly',  
    'age': 26,  
    'salary': 105000,  
    'city': 'New York'  
}
```

Try it!

You want to update the name of the key from 'city' to 'location' to better reflect your company's growing market share in non-urban areas.

Write Python code to do so and print the updated dictionary.

Solution



```
customer_info['location'] = customer_info['city']  
del customer_info['city']  
print(customer_info)
```

Can we write more efficient code?

Efficient Solution



```
customer_info['location'] = customer_info.pop('city')  
print(customer_info)
```


Try it!



You are given a list of strings as input and are required to return a dictionary whose keys are the unique two-letter suffixes of the input strings. The value of the dictionary to be returned is a list of all the strings that end with that two-letter suffix in the same order that they appear in. Write a function, `suffix_list`, that performs this.

For example, `suffix_list(['hello', 'melon', 'felon', 'fello', 'x', 'jimmy fallon'])` returns `{'lo': ['hello', 'fello'], 'on': ['melon', 'felon', 'jimmy fallon']}`.

Solution



```
def suffix_list(strs):  
    suffixes = {}  
    for s in strs:  
        if len(s) >= 2:  
            suffix = s[-2:]  
            if suffix not in suffixes:  
                suffixes[suffix] = []  
            suffixes[suffix].append(s)  
    return suffixes
```