# Cheat Sheet

### Declaring lists

name = [**value**] \* **length**

name = []

name = [value, value, ..., value]

Example:

numbers = [0] \* 10 # [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

numbers2 = [] # []

numbers3 = [18, 7, 1, -3, 29, 4]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *index* | *0* | *1* | *2* | *3* | *4* | *5* |
| *value* | 18 | 7 | 1 | -3 | 29 | 4 |

### Using lists

name[**index**] = **value** # setting

othername = name[index] # getting

Example:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *index* | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* |
| *value* | 0 | 0 | 0 | 42 | 0 | 0 | 0 | 23 | 0 | 0 |

numbers[3] = 42

numbers[7] = 23

|  |  |
| --- | --- |
| Lists as parameter Example:def average(**nums**): sum = 0 for i in range(len(nums)): sum += numbers[i] return sum / len(nums) | List as return value Example:def count\_digits(n): counts = [0] \* 10 while n > 0: counts[n % 10] += 1 n = n // 10 return counts |
| List traversals for i in range(len(list)): do something with list[i] ...Example:counts = [10, 30, 20, 4]sum = 0for i in range(len(counts)): sum += **counts[i]** | String traversals for i in range(len(string): do something with string[i] ...Example:phrase = "the quick brown fox"capital\_letters = 0for i in range(len(**phrase**)): letter = **phrase[i]** if letter >= 'A' && letter <= 'Z': capital\_letters += 1  |

### List functions

|  |  |
| --- | --- |
| **Functions** | **Description** |
| append(x) | Add an item to the end of the list. Equivalent to a[len(a):] = [x]. |
| extend(L) | Extend the list by appending all the items in the given list. Equivalent to a[len(a):] = L |
| insert(i, x) | Inserts an item at a given position. i is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list. |
| remove(x) | Removes the first item from the list whose value is x. Errs if there is no such item. |
| pop(i) | Removes the item at the given position in the list, and returns it. a.pop() removes and returns the last item in the list. |
| clear() | Remove all items from the list.  |
| index(x) | Returns the index in the list of the first item whose value is x. Errs if there is no such item. |
| count(x) | Returns the number of times x appears in the list. |
| sort() | Sort the items of the list |
| reverse() | Reverses the elements of the list |
| copy() | Return a copy of the list. |

# Questions

### List Simulation

1. Simulate the execution of the following function with each of the following lists passed as its parameter:

def mystery1(list):

 x = 0

 for i in range(1, len(list):

 y = list[i] - list[0]

 if y > x:

|  |
| --- |
| List Value returned[5] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[3, 12] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[4, 2, 10, 8] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[1, 9, 3, 5, 7] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[8, 2, 10, 4, 10, 9] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

 x = y

 return x

2. Simulate the execution of the following function with each of the following lists passed as its parameter:

def mystery2(nums):

 for i in range(0, len(nums) – 1):

 if nums[i] > nums[i + 1]:

 nums[i + 1] += 1

|  |
| --- |
| List Final list contents[8] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[14, 7] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[7, 1, 3, 2, 0, 4] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[10, 8, 9, 5, 5] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[12, 11, 10, 10, 8, 7] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

###

3. Simulate the execution of the following function with each of the following lists passed as its parameter:

def mystery3(data):

 for i in range(1, len(data) – 1):

 if data[i] == data[i - 1] + data[i + 1]:

 data[i] = data[i] // 2

|  |
| --- |
| List Final list contents[3, 7, 4] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[0, 3, 7, 4, 1] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[4, 3, 8, 5, 1, 2] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[2, 1, 5, 4, 10, 6, 2] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[1, 2, 1, 2, 1, 2, 1] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

###

 (continued on back page)

# Questions (continued)

### List Traversals

4. Write a function named find\_min that returns the minimum value in an list of integers. For example, if a list named list contains the values [16, 12, 25, 44], the call of find\_min(list) should return 12. You may assume that the list has at least one element.

5. Write a function named is\_sorted that accepts a list of floats as a parameter and returns True if the list is in sorted (nondecreasing) order and False otherwise. For example, if lists named list1 and list2 store [16.1, 12.3, 22.2, 14.4] and [1.5, 4.3, 7.0, 19.5, 25.1, 46.2] respectively, the calls is\_sorted(list1) and is\_sorted(list2) should return False and True respectively. Assume the list has at least one element. A one-element list is considered to be sorted.

6. Write a function remove\_even\_length that takes a list of strings as a parameter and that removes all of the strings of even length from the list. For example, if the list initially stores these values:

["to", "be", "or", "not", "to", "be", "hamlet"]

 After being passed to remove\_even\_length, it should store the following values:

["not"]

7. Write a function stutter that takes a list of strings as a parameter and that replaces every string with two of that string. For example, if the list stores the following values before the function is called:

["how", "are", "you?"]

 After the call, it should store the following values:

["how", "how", "are", "are", "you?", "you?"]

8. Write a function remove\_shorter\_strings that takes a list of strings as a parameter and that removes from each successive pair of values the shorter string in the pair. For example, suppose that a list called list contains the following values:

["four", "score", "and", "seven", "years", "ago"]

 In the first pair, "four" and "score", the shorter string is "four". In the second pair, "and" and "seven", the shorter string is "and". In the third pair, "years" and "ago", the shorter string is "ago". Therefore, the call remove\_shorter\_strings(list) should remove these, leaving the list as follows:

["score", "seven", "years"]

 If there is a tie (both strings have the same length), your function should remove the first string in the pair. If there is an odd number of strings in the list, the final value should be kept in the list.

### Lists for Counting and Tallying

9. Write a function named count\_last\_digits that accepts a list of integers as a parameter and examines its elements to determine how many end in 0, how many end in 1, how many end in 2 and so on. Your function will return a list of counters. The count of how many elements end in 0 should be stored in its element [0], how many of the values end in 1 should be stored in its element [1], and so on.

 For example, if a list named list contains the values [9, 29, 44, 103, 2, 52, 12, 12, 76, 35, 20], the call of count\_last\_digits(list) should return the list [1, 0, 4, 1, 1, 1, 1, 0, 0, 2] because 1 element ends with 0 (20), no elements end with 1, 4 elements end with 2 (2, 52, 12, and 12), etc.

10. Write a function named vowel\_count that accepts a String as a parameter and produces and returns a list of integers representing the counts of each vowel in the String. The list returned by your function should hold 5 elements: the first is the count of As, the second is the count of Es, the third Is, the fourth Os, and the fifth Us. Assume that the string contains no uppercase letters.

 For example, the call vowel\_count("i think, therefore i am") should return the list [1, 3, 3, 1, 0].

### 11. Write a list named sum5 that accepts an list of integers as a parameter and returns a new lsit of length 5 that contains a series of sums. The first value in the result should be the sum of every fifth number in the list starting with the first number (1st, 6th, 11th, 16th, etc), the second value in the result should be the sum of every fifth number in the list starting with the second number (2nd, 7th, 12th, 17th, etc), the third value in the result should be the sum of every fifth number in the list starting with the third number (3rd, 8th, 13th, 18th, etc), and so on.

### Advanced List Programming

12. Write a function named rotate\_right that accepts a list of integers as a parameter and rotates the values in the list to the right (i.e., forward in position) by one. Each element moves right by one, except the last element, which moves to the front. For example, if a variable named list refers to a list containing the values [3, 8, 19, 7], the call of rotate\_right(list) should modify it to store [7, 3, 8, 19]. A subsequent call of rotate\_right(list) would leave the list as follows: [19, 7, 3, 8]

13. Write a function named stretch that accepts a list of integers as a parameter and returns a new list twice as large as the original, replacing every integer from the original list with a pair of integers, each half the original. If a number in the original list is odd, then the first number in the new pair should be one higher than the second so that the sum equals the original number. For example, if a variable named list refers to an list storing the values [18, 7, 4, 24, 11], the call of stretch(list) should return a new list containing [9, 9, 4, 3, 2, 2, 12, 12, 6, 5]. (The number 18 is stretched into the pair 9, 9, the number 7 is stretched into 4, 3, the number 4 is stretched into 2, 2, the number 24 is stretched into 12, 12 and the number 11 is stretched into 6, 5.)

# Solutions

|  |  |  |
| --- | --- | --- |
| 1. | **List** | **Value Returned** |
|  | [5][3, 12][4, 2, 10, 8][1, 9, 3, 5, 7][8, 2, 10, 4, 10, 9] | 09682 |

|  |  |  |
| --- | --- | --- |
| 2. | **List** | **Final Contents** |
|  | [8][14, 7][7, 1, 3, 2, 0, 4][10, 8, 9, 5, 5][12, 11, 10, 10, 8, 7] | [8][14, 8][7, 2, 3, 3, 1, 4][10, 9, 9, 6, 6][12, 12, 11, 11, 9, 8] |

|  |  |  |
| --- | --- | --- |
| 3. | **List** | **Final Contents** |
|  | [3, 7, 4][0, 3, 7, 4, 1][4, 3, 8, 5, 1, 2][2, 1, 5, 4, 10, 6, 2][1, 2, 1, 2, 1, 2, 1] | [3, 3, 4][0, 3, 3, 2, 1][4, 3, 4, 2, 1, 2][2, 1, 2, 4, 5, 6, 2][1, 1, 1, 1, 1, 1, 1] |

4. (two solutions shown)

|  |  |
| --- | --- |
| def find\_min(list): min = list[0] for i in range(1, len(list)): if list[i] < min: min = list[i] return min | def find\_min(list): mini = list[0] for i in range(1, len(list)): mini = min(min, list[i]) return mini |

5.

def is\_sorted(list):

 for i in range(0, len(list) – 1):

 if list[i] > list[i + 1]:

 return False

 return True

6.

def remove\_even\_length(list):
 i = 0
 while i < len(list):
 if len(list[i]) % 2 == 0:
 list.remove(i)
 else:
 i += 1

7.

def stutter(list):

 i = 0
 while i < len(list):
 list.insert(i, list[i])

 i += 2

8.

def remove\_shorter\_strings(list):

 i = 0

 while i < len(list) - 1:
 first = list[i]
 second = list[i + 1]
 if len(first) <= len(second):
 list.remove(i)
 else:
 list.remove(i + 1)

 i += 1

9.

def count\_last\_digits(list):

 count = [0] \* 10

 for i in range(len(list)):

 digit = list[i] % 10

 count[digit] += 1

 return count

10. (two solutions shown)

def vowel\_count(text):

 counts = [0] \* 5

 for i in range(len(text)):

 c = text[i]

 if (c == 'a'):

 counts[0] += 1

 elif (c == 'e'):

 counts[1] += 1

 elif (c == 'i'):

 counts[2] += 1

 elif (c == 'o'):

 counts[3] += 1

 elif (c == 'u'):

 counts[4] += 1

 return counts

def vowel\_count(text):

 vowels = ['a', 'e', 'i', 'o', 'u']

 counts = [0] \* 5

 for i in range(len(text)):

 for j in range(0, len(vowels)):

 if text[i] == vowels[j]:

 counts[j] += 1

 return counts

11. (two solutions shown)

def sum5(list):

 result = [0] \* 5

 for i in range(0, len(list)):

 result[i % 5] += list[i]

 return result

def sum5(list):

 result = [0] \* 5

 for i in range(0, 5):

 for j in range(i, len(list)):

 result[i] += list[j]

 return result

12.

def rotate\_right(list):

 last = list[- 1]

 for j in range(list[-1], 0, -1):

 list[j] = list[j - 1]

 list[0] = last

13.

def stretch(list):

 result = [0] \* (2 \* len(list))

 for i in range(0, len(list)):

 result[2 \* i] = list[i] // 2 + list[i] % 2

 result[2 \* i + 1] = list[i] // 2

 return result