

# Python Cheat Sheet: 14 Interview Questions

"A puzzle a day to learn, code, and play" →

\*FREE\* Python Email Course @ <http://bit.ly/free-python-course>

| Question  | Code   | Question                                     | Code   |
|---|--|--|--|
| <b>Check if list contains integer x</b>                                       | <pre>l = [3, 3, 4, 5, 2, 111, 5] print(111 in l) # True</pre>  | <b>Get missing number in [1...100]</b>       | <pre>def get_missing_number(lst):     return set(range(lst[len(lst)-1])[1:]) - set(l) l = list(range(1,100)) l.remove(50) print(get_missing_number(l)) # 50</pre>  |
| <b>Find duplicate number in integer list</b>                                  | <pre>def find_duplicates(elements):     duplicates, seen = set(), set()     for element in elements:         if element in seen:             duplicates.add(element)             seen.add(element)     return list(duplicates)</pre> | <b>Compute the intersection of two lists</b> | <pre>def intersect(lst1, lst2):     res, lst2_copy = [], lst2[:]     for el in lst1:         if el in lst2_copy:             res.append(el)             lst2_copy.remove(el)     return res</pre>  |
| <b>Check if two strings are anagrams</b>                                      | <pre>def is_anagram(s1, s2):     return set(s1) == set(s2) print(is_anagram("elvis", "lives")) # True</pre>  | <b>Find max and min in unsorted list</b>     | <pre>l = [4, 3, 6, 3, 4, 888, 1, -11, 22, 3] print(max(l)) # 888 print(min(l)) # -11</pre>   |
| <b>Remove all duplicates from list</b>  | <pre>lst = list(range(10)) + list(range(10)) lst = list(set(lst)) print(lst) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]</pre>  | <b>Reverse string using recursion</b>        | <pre>def reverse(string):     if len(string)&lt;=1: return string     return reverse(string[1:])+string[0] print(reverse("hello")) # olleh</pre>   |
| <b>Find pairs of integers in list so that their sum is equal to integer x</b> | <pre>def find_pairs(l, x):     pairs = []     for (i, el_1) in enumerate(l):         for (j, el_2) in enumerate(l[i+1:]):             if el_1 + el_2 == x:                 pairs.append((el_1, el_2))     return pairs</pre>         | <b>Compute the first n Fibonacci numbers</b> | <pre>a, b = 0, 1 n = 10 for i in range(n):     print(b)     a, b = b, a+b # 1, 1, 2, 3, 5, 8, ...</pre>  |
| <b>Check if a string is a palindrome</b>                                      | <pre>def is_palindrome(phrase):     return phrase == phrase[::-1] print(is_palindrome("anna")) # True</pre>  | <b>Sort list with Quicksort algorithm</b>    | <pre>def qsort(L):     if L == []: return []     return qsort([x for x in L[1:] if x&lt; L[0]]) + L[0:1] +     qsort([x for x in L[1:] if x&gt;=L[0]]) lst = [44, 33, 22, 5, 77, 55, 999] print(qsort(lst)) # [5, 22, 33, 44, 55, 77, 999]</pre>   |
| <b>Use list as stack, array, and queue</b>                                    | <pre># as a list ... l = [3, 4] l += [5, 6] # l = [3, 4, 5, 6]  # ... as a stack ... l.append(10) # l = [4, 5, 6, 10] l.pop() # l = [4, 5, 6]  # ... and as a queue l.insert(0, 5) # l = [5, 4, 5, 6] l.pop() # l = [5, 4, 5]</pre>  | <b>Find all permutation s of string</b>      | <pre>def get_permutations(w):     if len(w)&lt;=1:         return set(w)     smaller = get_permutations(w[1:])     perms = set()     for x in smaller:         for pos in range(0,len(x)+1):             perm = x[:pos] + w[0] + x[pos:]             perms.add(perm)     return perms print(get_permutations("nan")) # {'nna', 'ann', 'nan'}</pre> |

