Python Cheat Sheet: 14 Interview Questions

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

Question	Code	Question	Code
| **Check if list contains integer x** | ```python
l = [3, 3, 4, 5, 2, 111, 5]
print(111 in l) # True
``` | **Get missing number in [1...100]** | ```python
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
``` |
| **Find duplicate number in integer list** | ```python
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)
``` | **Compute the intersection of two lists** | ```python
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
``` |
| **Check if two strings are anagrams** | ```python
def is_anagram(s1, s2):
    return set(s1) == set(s2)
print(is_anagram("elvis", "lives")) # True
``` | **Find max and min in unsorted list** | ```python
l = [4, 3, 6, 3, 4, 888, 1, -11, 22, 3]
print(max(l)) # 888
print(min(l)) # -11
``` |
| **Remove all duplicates from list** | ```python
lst = list(range(10)) + list(range(10))
lst = list(set(lst))
print(lst)
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
``` | **Reverse string using recursion** | ```python
def reverse(string):
    if len(string)<=1: return string
    return reverse(string[1:])+string[0]
print(reverse("hello")) # olleh
``` |

Python Cheat Sheet: 14 Interview Questions

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

Find pairs of integers in list so that their sum is equal to integer x	```python\ndef find_pairs(l, x):\n pairs = []\n for (i, el_1) in enumerate(l):\n for (j, el_2) in enumerate(l[i+1:]):\n if el_1 + el_2 == x:\n pairs.append((el_1, el_2))\n return pairs\n```	Compute the first n Fibonacci numbers	```python\na, b = 0, 1\nn = 10\nfor i in range(n):\n print(b)\n a, b = b, a+b\n# 1, 1, 2, 3, 5, 8, ...\n```
Check if a string is a palindrome	```python\ndef is_palindrome(phrase):\n return phrase == phrase[::-1]\nprint(is_palindrome("anna")) # True\n```	Sort list with Quicksort algorithm	```python\ndef qsort(L):\n if L == []: return []\n return qsort([x for x in L[1:] if x< L[0]]) + L[0:1] +\nqsort([x for x in L[1:] if x>=L[0]])\nlst = [44, 33, 22, 5, 77, 55, 999]\nprint(qsort(lst))\n# [5, 22, 33, 44, 55, 77, 999]\n```
Use list as stack, array, and queue	```python\n# as a list ...\nl = [3, 4]\nl += [5, 6] # l = [3, 4, 5, 6]\n\n# ... as a stack ...\nl.append(10) # l = [4, 5, 6, 10]\nl.pop() # l = [4, 5, 6]\n\n# ... and as a queue\nl.insert(0, 5) # l = [5, 4, 5, 6]\nl.pop() # l = [5, 4, 5]\n```	Find all permutations of string	```python\ndef get_permutations(w):\n if len(w)<=1:\n return set(w)\n smaller = get_permutations(w[1:])\n perms = set()\n for x in smaller:\n for pos in range(0,len(x)+1):\n perm = x[:pos] + w[0] + x[pos:]\n perms.add(perm)\n return perms\nprint(get_permutations("nan"))\n# {'nna', 'ann', 'nan'}\n```