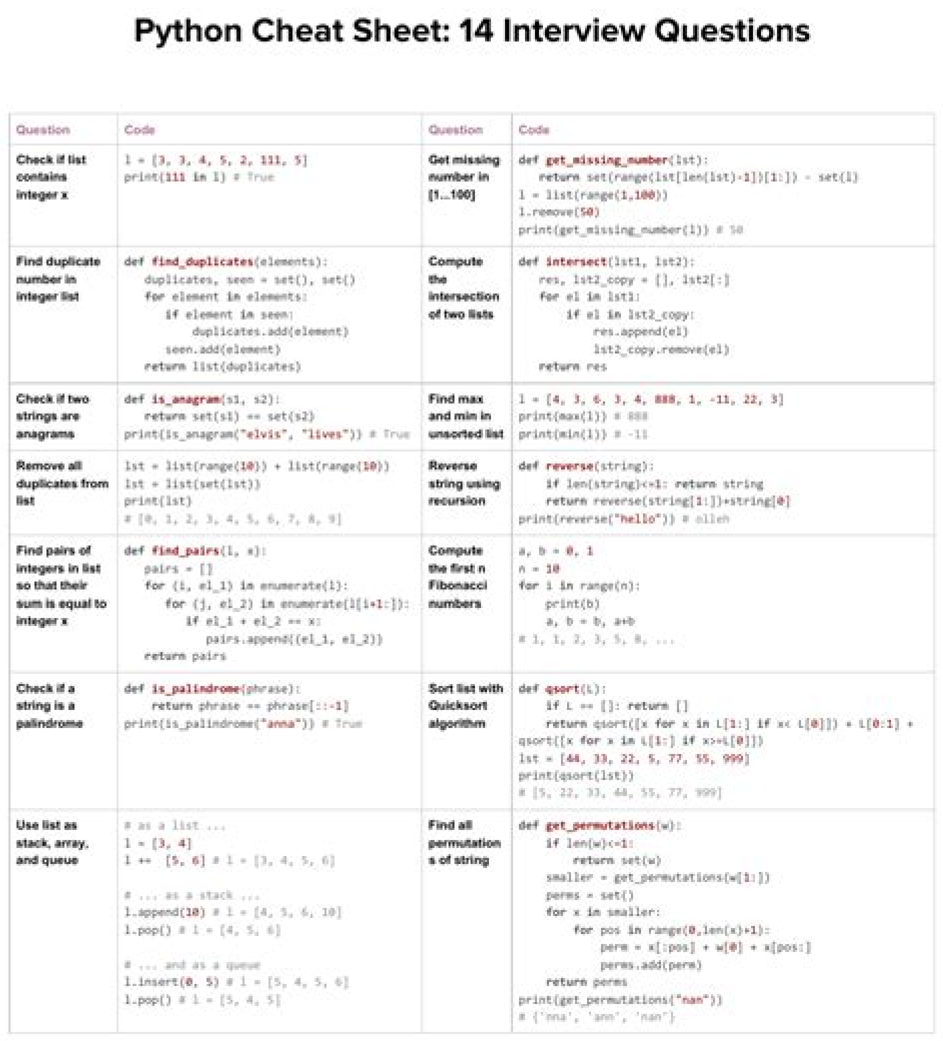
Python Coding Interview Cheat Sheet



Python Cheat Sheet: 14 Interview Questions

Question	Code	Question	Code

Check if list contains integer x	`l = [3, 3, 4, 5, 2, 111, 5]`
	`print(111 in l) # True`
		Get missing number in [1...100]	`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	`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	`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	`def is_anagram(s1, s2):`
	`    return set(s1) == set(s2)`
	`print(is_anagram("elvis", "lives")) # True`
		Find max and min in unsorted list	`l = [4, 3, 6, 3, 4, 888, 1, -11, 22, 3]`
			`print(max(l)) # 888`
			`print(min(l)) # -11`

Remove all duplicates from list	`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	`def reverse(string):`
			`    if len(string)<=1: return string`
			`    return reverse(string[1:])+string[0]`
			`print(reverse("hello")) # olleh`

Find pairs of integers in list so that their sum is equal to integer x	`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`
		Compute the first n Fibonacci numbers	`a, b = 0, 1`
			`n = 10`
			`for i in range(n):`
			`    print(b)`
			`    a, b = b, a+b`
			`# 1, 1, 2, 3, 5, 8, ...`

Check if a string is a palindrome	`def is_palindrome(phrase):`
	`    return phrase == phrase[::-1]`
	`print(is_palindrome("anna")) # True`
		Sort list with Quicksort algorithm	`def qsort(L):`
			`    if L == []: return []`
			`    return qsort([x for x in L[1:] if x< L[0]]) + L[0:1] +`
			`qsort([x for x in L[1:] if x>=L[0]])`
			`lst = [44, 33, 22, 5, 77, 55, 999]`
			`print(qsort(lst))`
			`# [5, 22, 33, 44, 55, 77, 999]`

Use list as stack, array, and queue	`# 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]`
		Find all permutations of string	`def get_permutations(w):`
			`    if len(w)<=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'}`

Python coding interview cheat sheet is an essential resource for software developers preparing for interviews that focus on Python programming skills. In today's competitive job market, having a solid grasp of Python concepts, data structures, algorithms, and best practices can significantly enhance a candidate's chances of success. This article serves as a comprehensive guide, summarizing critical topics, techniques, and common questions to help you prepare effectively for Python coding interviews.

1. Python Basics

Understanding the fundamentals of Python is crucial. Here are the key topics you should master:

1.1 Data Types

Python has several built-in data types:

- Integers: Whole numbers, e.g., `x = 10`
- Floats: Decimal numbers, e.g., `y = 3.14`
- Strings: Text data, e.g., `name = "Alice"`
- Lists: Ordered collections, e.g., `fruits = ["apple", "banana"]`
- Tuples: Immutable collections, e.g., `coordinates = (10.0, 20.0)`
- Dictionaries: Key-value pairs, e.g., `person = {"name": "John", "age": 30}`
- Sets: Unordered collections of unique elements, e.g., `unique_numbers = {1, 2, 3}`

1.2 Control Structures

Control structures dictate the flow of a program. Key constructs include:

- Conditionals: `if`, `elif`, and `else`
- Loops: `for` loops and `while` loops
- Comprehensions: List, dictionary, and set comprehensions for concise data manipulation

1.3 Functions

Functions are reusable blocks of code. Key concepts include:

- Defining Functions: Using the `def` keyword
- Parameters and Return Values: Passing arguments and returning results
- Lambda Functions: Anonymous functions, e.g., `add = lambda x, y: x + y`
- Scope and Lifetime: Understanding local vs. global variables

2. Data Structures

A solid grasp of data structures is vital for optimizing algorithms and solving complex problems.

2.1 Lists

- Operations: Accessing elements, slicing, appending, removing
- Common Methods:
- `append()`
- `insert()`
- `remove()`
- `pop()`
- `sort()`

2.2 Dictionaries

- Key Operations: Accessing, adding, and removing key-value pairs
- Common Methods:
- `keys()`
- `values()`
- `items()`

2.3 Sets

- Properties: Uniqueness, unordered nature
- Common Operations: Union, intersection, difference

2.4 Tuples

- Immutability: Why tuples are useful
- Common Use Cases: Returning multiple values from functions

3. Algorithms

Understanding algorithms is crucial for problem-solving in coding interviews.

3.1 Sorting Algorithms

- Bubble Sort: Simple, but inefficient
- Merge Sort: Efficient, divide-and-conquer strategy
- Quick Sort: Efficient, uses pivoting
- Python Built-in Sort: Using `sorted()` and `list.sort()`

3.2 Searching Algorithms

- Linear Search: Simple but inefficient for large datasets
- Binary Search: Efficient for sorted lists, requires O(log n) complexity

3.3 Time and Space Complexity

- Big O Notation: Understanding how to analyze algorithm efficiency
- Common Complexities:
- O(1): Constant time
- O(n): Linear time
- O(n^2): Quadratic time

4. Object-Oriented Programming (OOP)

OOP principles are fundamental in Python. Key concepts include:

4.1 Classes and Objects

- Defining Classes: Using the `class` keyword
- Creating Objects: Instantiating classes

4.2 Inheritance

- Single Inheritance: One class inherits from another
- Multiple Inheritance: A class inherits from multiple classes

4.3 Encapsulation

- Private and Public Attributes: Using underscores to denote private variables
- Getters and Setters: Accessing private attributes through methods

4.4 Polymorphism

- Method Overriding: Changing the behavior of inherited methods
- Method Overloading: Defining multiple methods with the same name

5. Common Coding Interview Questions

Practicing common coding problems can significantly enhance your problem-solving skills.

5.1 String Manipulation

- Reverse a String: Using slicing or a loop
- Check for Anagrams: Comparing sorted versions or using `collections.Counter`

5.2 Array Problems

- Two Sum Problem: Finding two numbers that add up to a target
- Find the Missing Number: Using arithmetic or set operations

5.3 Linked Lists

- Reverse a Linked List: Iterative and recursive approaches
- Detect Cycle: Using Floyd’s Tortoise and Hare algorithm

5.4 Trees and Graphs

- Binary Tree Traversal: Pre-order, in-order, post-order
- Finding Height of a Tree: Recursive depth-first approach

5.5 Dynamic Programming

- Fibonacci Sequence: Using memoization
- Knapsack Problem: Optimizing resource allocation

6. Best Practices

To stand out in coding interviews, follow these best practices:

- Write Clean Code: Use meaningful variable names and consistent formatting.
- Comment Your Code: Explain complex logic and algorithms.
- Test Your Code: Check edge cases and use assertions.
- Optimize: Always look for ways to reduce time and space complexity.
- Communicate: Explain your thought process while coding to the interviewer.

7. Resources for Further Study

To prepare thoroughly for your Python coding interview, consider the following resources:

- Books:
- "Cracking the Coding Interview" by Gayle Laakmann McDowell
- "Python Crash Course" by Eric Matthes

- Online Platforms:
- LeetCode: Practice coding problems in Python
- HackerRank: Participate in challenges and improve your skills
- Codecademy: Interactive Python courses

- Documentation:
- Official Python Documentation: Comprehensive resource for Python features and functions

In conclusion, leveraging a Python coding interview cheat sheet can simplify your preparation process and enhance your confidence during interviews. By mastering the outlined topics and practicing coding problems, you can significantly increase your chances of success in securing a coveted Python development position. Remember, consistent practice and thorough understanding are key to excelling in technical interviews. Good luck!

Frequently Asked Questions

What is a Python coding interview cheat sheet?

A Python coding interview cheat sheet is a concise reference guide that summarizes important Python concepts, functions, and coding patterns often tested in technical interviews.

What topics should be included in a Python coding interview cheat sheet?

Key topics include data structures (lists, dictionaries, sets), algorithms (sorting, searching), object-oriented programming, exception handling, and common libraries like NumPy and Pandas.

How can I effectively use a Python coding interview cheat sheet?

Use the cheat sheet as a quick reference before the interview to refresh your memory on key concepts, and practice coding problems that utilize those concepts.

Are there any recommended resources for creating a Python coding interview cheat sheet?

Yes, resources like 'LeetCode', 'GeeksforGeeks', and 'Cracking the Coding Interview' provide valuable information and examples to help compile an effective cheat sheet.

What are some common Python interview questions that can be summarized in a cheat sheet?

Common questions include: 'How do you reverse a string?', 'What are list comprehensions?', and 'How do you handle exceptions in Python?'.

Is it beneficial to memorize the cheat sheet or understand the concepts?

While memorizing can help, it's more important to understand the underlying concepts so you can apply them flexibly during problem-solving in interviews.

Can a Python coding interview cheat sheet help with system design interviews?

While primarily focused on coding, a Python cheat sheet can assist in system design interviews by providing insights into Python's capabilities for handling data and scalability.

Python Coding Interview Cheat Sheet

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm ...

What does asterisk * mean in Python? - Stack Overflow
What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the "at" (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does ...

Is there a "not equal" operator in Python? - Stack Overflow
Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library ...

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and ...

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the ...

python - Is there a difference between "==" and "is"? - Stack ...
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows ...

python - What does ** (double star/asterisk) and * (star/asterisk) ...
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion ...

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm ...

What does asterisk * mean in Python? - Stack Overflow
What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the “at” (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does ...

Is there a "not equal" operator in Python? - Stack Overflow
Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library ...

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and ...

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the ...

python - Is there a difference between "==" and "is"? - Stack ...
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows ...

python - What does ** (double star/asterisk) and * (star/asterisk) ...
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion ...

Ace your next coding interview with our comprehensive Python coding interview cheat sheet. Discover essential tips and resources to boost your confidence. Learn more!

Back to Home