Python Interview Handbook — Batch 10

Generated: 2025-09-13 02:00:06Z (UTC)

Python Theory & Cheatsheet

```
PYTHON FUNDAMENTALS & CHEATSHEET
--------------------------------
- Data Types: int, float, str, bool, list, tuple, dict, set
- Comprehensions: [x for x in ...], {k:v for ...}, {x for ...}
- Functions: def, *args, **kwargs, closures
- OOP: classes, inheritance, dunder methods (__init__, __repr__)
- Decorators: @decorator, wraps; Context Managers: with, __enter__/__exit__
- Errors: try/except/else/finally; raise
- Iterators & Generators: iter(), next(), yield
- Useful libs: itertools, functools, collections, heapq, bisect
- Tips: enumerate, zip, sorted key=, slicing, unpacking
```

091. Serialize Graph Adjlist

Statement: Implement problem “serialize graph adjlist” in Python.
Explanation: Problem “serialize graph adjlist”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def serialize_graph_adjlist(*args, **kwargs):
    """TODO: implement serialize_graph_adjlist as described in the handbook."""
    return None
```

```
import unittest
from problems.serialize_graph_adjlist import serialize_graph_adjlist
class TestSerializeGraphAdjlist(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(serialize_graph_adjlist())
```

092. Deserialize Graph Adjlist

Statement: Implement problem “deserialize graph adjlist” in Python.
Explanation: Problem “deserialize graph adjlist”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def deserialize_graph_adjlist(*args, **kwargs):
    """TODO: implement deserialize_graph_adjlist as described in the handbook."""
    return None
```

```
import unittest
from problems.deserialize_graph_adjlist import deserialize_graph_adjlist
class TestDeserializeGraphAdjlist(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(deserialize_graph_adjlist())
```

093. Palindrome Partitioning

Statement: Implement problem “palindrome partitioning” in Python.
Explanation: Problem “palindrome partitioning”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def palindrome_partitioning(*args, **kwargs):
    """TODO: implement palindrome_partitioning as described in the handbook."""
    return None
```

```
import unittest
from problems.palindrome_partitioning import palindrome_partitioning
class TestPalindromePartitioning(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(palindrome_partitioning())
```

094. Min Window Substring

Statement: Implement problem “min window substring” in Python.
Explanation: Problem “min window substring”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def min_window_substring(*args, **kwargs):
    """TODO: implement min_window_substring as described in the handbook."""
    return None
```

```
import unittest
from problems.min_window_substring import min_window_substring
class TestMinWindowSubstring(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(min_window_substring())
```

095. Max Rectangle Histogram

Statement: Implement problem “max rectangle histogram” in Python.
Explanation: Problem “max rectangle histogram”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def max_rectangle_histogram(*args, **kwargs):
    """TODO: implement max_rectangle_histogram as described in the handbook."""
    return None
```

```
import unittest
from problems.max_rectangle_histogram import max_rectangle_histogram
class TestMaxRectangleHistogram(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(max_rectangle_histogram())
```

096. Sliding Window Max

Statement: Implement problem “sliding window max” in Python.
Explanation: Problem “sliding window max”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def sliding_window_max(*args, **kwargs):
    """TODO: implement sliding_window_max as described in the handbook."""
    return None
```

```
import unittest
from problems.sliding_window_max import sliding_window_max
class TestSlidingWindowMax(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(sliding_window_max())
```

097. Design Hashmap Simple

Statement: Implement problem “design hashmap simple” in Python.
Explanation: Problem “design hashmap simple”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def design_hashmap_simple(*args, **kwargs):
    """TODO: implement design_hashmap_simple as described in the handbook."""
    return None
```

```
import unittest
from problems.design_hashmap_simple import design_hashmap_simple
class TestDesignHashmapSimple(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(design_hashmap_simple())
```

098. Design Cache Ttl

Statement: Implement problem “design cache ttl” in Python.
Explanation: Problem “design cache ttl”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def design_cache_ttl(*args, **kwargs):
    """TODO: implement design_cache_ttl as described in the handbook."""
    return None
```

```
import unittest
from problems.design_cache_ttl import design_cache_ttl
class TestDesignCacheTtl(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(design_cache_ttl())
```

099. Url Validation

Statement: Implement problem “url validation” in Python.
Explanation: Problem “url validation”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def url_validation(*args, **kwargs):
    """TODO: implement url_validation as described in the handbook."""
    return None
```

```
import unittest
from problems.url_validation import url_validation
class TestUrlValidation(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(url_validation())
```

100. Email Validation Regex

Statement: Implement problem “email validation regex” in Python.
Explanation: Problem “email validation regex”. Outline brute-force vs optimized approaches, edge cases, and complexity.

```
def email_validation_regex(*args, **kwargs):
    """TODO: implement email_validation_regex as described in the handbook."""
    return None
```

```
import unittest
from problems.email_validation_regex import email_validation_regex
class TestEmailValidationRegex(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(email_validation_regex())
```