Python Interview Handbook — Batch 8

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
```

071. Parse Csv Line

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

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

```
import unittest
from problems.parse_csv_line import parse_csv_line
class TestParseCsvLine(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(parse_csv_line())
```

072. Url Shortener Encode Decode

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

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

```
import unittest
from problems.url_shortener_encode_decode import url_shortener_encode_decode
class TestUrlShortenerEncodeDecode(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(url_shortener_encode_decode())
```

073. Rate Limiter Fixed Window

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

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

```
import unittest
from problems.rate_limiter_fixed_window import rate_limiter_fixed_window
class TestRateLimiterFixedWindow(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(rate_limiter_fixed_window())
```

074. Token Bucket Limiter

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

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

```
import unittest
from problems.token_bucket_limiter import token_bucket_limiter
class TestTokenBucketLimiter(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(token_bucket_limiter())
```

075. Debounce Function

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

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

```
import unittest
from problems.debounce_function import debounce_function
class TestDebounceFunction(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(debounce_function())
```

076. Throttle Function

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

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

```
import unittest
from problems.throttle_function import throttle_function
class TestThrottleFunction(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(throttle_function())
```

077. Decorator Timing

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

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

```
import unittest
from problems.decorator_timing import decorator_timing
class TestDecoratorTiming(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(decorator_timing())
```

078. Context Manager Example

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

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

```
import unittest
from problems.context_manager_example import context_manager_example
class TestContextManagerExample(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(context_manager_example())
```

079. Pickle Serialize

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

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

```
import unittest
from problems.pickle_serialize import pickle_serialize
class TestPickleSerialize(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(pickle_serialize())
```

080. Json Serialize Custom

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

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

```
import unittest
from problems.json_serialize_custom import json_serialize_custom
class TestJsonSerializeCustom(unittest.TestCase):
    def test_placeholder(self):
        self.assertIsNone(json_serialize_custom())
```