Python: Data Science and ML Refresher

Shubhankar Agrawal

Abstract

This document serves as a quick refresher for Data Science and Machine Learning interviews. It covers Python concepts required for Machine Learning and Data Science. This requires the reader to have a foundational level knowledge with tertiary education in the field. This PDF contains material for revision over key concepts that are tested in interviews.

Contents

1 Python Fundamentals 1

1.1 Variables . . . 1

1.2 Data Types . . . 1
Primitives • Non-Primitives

1.3 Concepts . . . 1
Typing • Object Reference • Evaluation • Garbage Collection • Styling

2 Data Structures and Algorithms 1

2.1 Data Structures . . . 1

2.2 Variants . . . 2
Tuple • List • Set • Dictionary

2.3 Algorithms . . . 2

3 Object Oriented Programming 2

4 Advanced Topics 2

4.1 Pythonic Functionalities . . . 2
List Comprehension • Lambda • Context Manager • Decorators

4.2 Control Flow . . . 3
Iterating • Generating

4.3 Concurrency . . . 3

4.4 Type Hinting . . . 3

5 Libraries 3

6 Contact me 3

1. Python Fundamentals

Interpreted language. Code converted to bytecode using interpreter (CPython default).

1.1. Variables

Object references. The `id` function is used to get the object identifier. Values $[-5, 256]$ cached on startup.

Table 1. Object Headers (Overhead)

	Bytes	Description	Notes
ob_refcnt	8	Reference Count	
ob_type	8	Pointer to Class	
ob_size	8	# Elements	Variable Length

Mutability: Object can be changed => Hashable

1.2. Data Types

1.2.1. Primitives

Floating Point Precision

1.2.2. Non-Primitives

Most of these are collection-based objects. Require reallocation when memory exceeds.

Tuple: Immutable.
List: Indices map to memory hashes. Mutable.
Set: HashSet. Mutable.
Dictionary: HashMap. Mutable.

Table 2. Primitive Data Types

Type	Bytes (Header)	Example	Reference
int	4 (24)	42	Variable size
float	8 (16)	12.46	C Double
bool	4 (24)	True	Integer
complex	16 (16)	True	Two Floats
bytes	n (24)	b01	Variable size

Table 3. Primitive Data Types

Type	Bits	Sign	Exponent	Significant
Single	32	1	8	23
Double (default)	64	1	11	52

System Configuration

All experiments have been run on:
System: Windows 64 bit
Python: 3.10

1.3. Concepts

1.3.1. Typing

Strong v Weak: Strong means type matters during operations (Cannot add str to int).

Static v **Dynamic**: Types can change in runtime (object of str can be reassigned to int).

1.3.2. Object Reference

Mutable objects are call by reference, immutable objects are call by value.

Use `nonlocal` keyword to reference variable outside function (inside module), and `global` keyword for global variable in script.

1.3.3. Evaluation

Eager: Evaluate complete function.
Lazy: Evaluate only what is necessary.

1.3.4. Garbage Collection

Objects are destroyed when 0 references (Strong vs Weak).

1.3.5. Styling

Python Enhancement Proposal 8 **(PEP8)** is a comprehensive style guide for Python.

Casing:

- camelCase
- PascalCase
- snake_case

2. Data Structures and Algorithms

2.1. Data Structures

Python implements data structures of tuples, lists, sets and dictionaries by default.

Table 4. Non-Primitive Data Types

Type	Bytes (Header)	Object	Additional
bytearray	32 (24)	b”	1
tuple	16 (24)	(a,b)	8
list	32 (24)	[a,b]	8
set	192 (24)	a,b	Hash Table
dict	208 (24)	a:b	16 + Hash Table

heapq: Min heap. `logn` complexities.
bintrees.FastRBTree: Red Black Tree.
networkx.Graph: Graph with Nodes and Edges
Thread safe queue libraries:

- queue.Queue
- queue.LIFOQueue
- queue.PriorityQueue

Table 5. Time Complexity - Big O

Function	Tuple	List	Set	Dictionary
x in s	n	n	1	1
Get Item	1	1	-	1
Append Item	-	1	1	1
Delete Item	-	n	1	1

2.2. Variants

2.2.1. Tuple

Variants of tuples with added functionalities.

collections.namedtuple: Access elements by name.

dataclasses.dataclass: Class decorator. Mutable.

numpy.recarray: Variant of ndarray with named fields.

2.2.2. List

Variants of lists with added functionalities.

collections.deque: Double ended queue implemented as a doubly linked list.

numpy.array: List with single data type. Memory efficient.

numpy.ndarray: Multidimensional array with vectorized operations.

pandas.Series: List with custom index mapping to each element.

2.2.3. Set

Variants of sets with added functionalities.

frozenset: Immutable set.

blist.sortedset: Maintains elements in sorted order with tree.

2.2.4. Dictionary

Variants of dictionaries with added functionalities.

collections.OrderedDict: Maintains order of insertion. Default in Python 3.

collections.defaultdict: Returns default value if missing.

collection.Counter: Frequency dictionary.

collections.ChainMap: Maintains update order of elements across dictionaries.

frozendict: Immutable dict.

blist.sorteddict: Maintains elements in sorted order with tree.

pandas.DataFrame: Two dimensional tabular data.

Variants of dicts, tuples, queues (collections)

Table 6. Sorting

Name	Big-O Time	Space
Tim (default)	$n \cdot \log n$	n
Bubble	n^2	1
Insertion	n^2	1
Selection	n^2	1
Merge	$n \cdot \log n$	n
Quick	n^2	$\log n$
Heap	$n \cdot \log n$	1
Count	$n + k$	n
Radix	$n \cdot k$	$n + k$

Table 7. Graph

Name	Big-O Time	Purpose
DFS	$V + E$	Traverse Graph
BFS	$V + E$	Traverse Graph
Dijkstra	$(V + E) \log V$	Shortest path S -> All Nodes
Bellman-Ford	VE	Shortest paths (- weights)
Floyd-Warshall	V^3	Shortest paths (All vertices)
Kruskal	$E \log E$	Minimum Spanning Tree
Prim	$(V + E) \log V$	MST from arbitrary node
Topologic Sort	$V + E$	Order DAG forward edges
Tarjan	$V + E$	Strongly Connected Comps
A* Search	$E \log V$	Shortest path + heuristics
Union-Find	$\alpha(V)$	Merge connected comps

2.3. Algorithms

Common Algorithms to be known and their time complexities

Binary Search ($\log n$) more efficient than Linear.

Dynamic Programming

- Fibonacci
- Knapsack (0-1, Repeated, Double Knapsack)
- Longest Common Subsequence
- Longest Increasing Subsequence
- Coin Change
- Edit Distance (Levenshtein)

Other algorithms to be familiar with:

- Huffman Encoding
- N Queens
- Non Overlapping Activities
- Subset sum
- Trie Build and Search
- Fast Exponentiation
- Sliding Window contiguous subsequence problems
- Reservoir Sampling
- Bit manipulation (AND &, OR |, XORˆ)

3. Object Oriented Programming

Classes form a key structure in Python.

4. Advanced Topics

4.1. Pythonic Functionalities

Here are some functionalities that help in writing mintainable Python code.

4.1.1. List Comprehension

Concise, clear **Pythonic** implementation of loops.

Example: `[x[:3] for x in items if type(x) == 'str']`

4.1.2. Lambda

Anonymous functions to be used within modules.

Example: `lambda x: x+1`

4.1.3. Context Manager

Safely open close and operate with files. Uses the `with` keyword.

4.1.4. Decorators

Add additional functionality to a function wrapping with more code.

```
# Using the decorator with a custom message
@log_function_calls("Logging")
def say_hello(name):
    print(f"Hello, {name}!")
```

Code 1. Decorator Usage

```
def log_function_calls(msg):
    def decorator(fn):
        def wrapper(*args, **kwargs):
            print(f"{msg}: {fn.__name__}.")
            return func(*args, **kwargs)
        return wrapper
    return decorator
```

Code 2. Decorator

4.2. Control Flow

- if, elif, else
- for
- continue, break
- match case

4.2.1. Iterating

Iterating and Generating

Iterator: Object that allows you to traverse through a collection.

Iterable: Collection that returns iterator

```
iterable = [1, 2, 3]
iterator = iter(iterable)
print(next(iterator))  # 1
```

Code 3. Iterating

4.2.2. Generating

Yield values one at a time. Maintain state. Lazy evaluation.

```
def my_generator():
    yield 1
    yield 2
    yield 3

gen = my_generator()
print(next(gen))  # 1
```

Code 4. Generating

4.3. Concurrency

The Global Interpreter Lock (GIL) prevents multithreading in Python

Concurrency, Asynchronous, Multithreading

4.4. Type Hinting

5. Libraries

Relevant Python libraries for Data Science / Machine Learning Roles

Table 8. Libraries

Type	Name	Purpose
General	json time datetime	JSON conversions Time profiling datetime parsing
DSA	heapq collections	Heaps Collections

6. Contact me

You can contact me through these methods:

🔗 Personal Website - astronights.github.io
✉ shubhankar.31@gmail.com
in linkedin.com/in/shubhankar-agrawal