50 Python Questions: Strings, Lists, Tuples & Dictionaries

Strings (Questions 1-15)

1. How do you create and access characters in a string?

python

```python
# Creating strings
text = "Hello World"
single_quote = 'Python'
multiline = """This is a
multiline string"""

# Accessing characters
print(text[0])      # H
print(text[-1])     # d
print(text[1:5])    # ello
```

2. How do you concatenate and repeat strings?

python

```python
str1 = "Hello"
str2 = "World"

# Concatenation
result1 = str1 + " " + str2        # Hello World
result2 = f"{str1} {str2}"         # Hello World
result3 = "{} {}".format(str1, str2)  # Hello World

# Repetition
repeated = "Python" * 3            # PythonPythonPython
```

3. How do you find the length and check membership in strings?

python

```python
text = "Python Programming"

# Length
length = len(text)                  # 18

# Membership
contains_python = "Python" in text # True
contains_java = "Java" in text     # False
starts_with = text.startswith("Py") # True
ends_with = text.endswith("ing")   # True
```

4. How do you use string methods for case conversion?

python

```python
text = "Hello World"

# Case methods
upper_text = text.upper()           # HELLO WORLD
lower_text = text.lower()           # hello world
title_text = text.title()           # Hello World
capitalize_text = text.capitalize() # Hello world
swapcase_text = text.swapcase()     # hELLO wORLD
```

5. How do you split and join strings?

python

```python
sentence = "apple,banana,orange,grape"
words = "Hello World Python"

# Split
fruits = sentence.split(",")       # ['apple', 'banana', 'orange', 'grape']
word_list = words.split()          # ['Hello', 'World', 'Python']
word_list2 = words.split(" ", 1)   # ['Hello', 'World Python']

# Join
joined = "-".join(fruits)          # apple-banana-orange-grape
spaced = " ".join(['a', 'b', 'c']) # a b c
```

6. How do you replace and strip strings?

```python
text = "  Hello World Hello  "

# Replace
replaced = text.replace("Hello", "Hi")        # "  Hi World Hi  "
replaced_once = text.replace("Hello", "Hi", 1) # "  Hi World Hello  "

# Strip
stripped = text.strip()                       # "Hello World Hello"
left_stripped = text.lstrip()                 # "Hello World Hello  "
right_stripped = text.rstrip()                # "  Hello World Hello"
char_stripped = "###Hello###".strip("#")      # "Hello"
```

7. How do you find and count substrings?

```python
text = "Python is awesome and Python is powerful"

# Find
index1 = text.find("Python")       # 0 (first occurrence)
index2 = text.find("Python", 10)   # 23 (search from index 10)
index3 = text.find("Java")         # -1 (not found)

# Count
count = text.count("Python")       # 2
count_is = text.count("is")        # 2

# Index (raises exception if not found)
try:
    index4 = text.index("awesome")  # 10
except ValueError:
    print("Not found")
```

8. How do you check string properties with is methods?

python

```python
# Various string checks
print("123".isdigit())      # True
print("abc".isalpha())      # True
print("abc123".isalnum())   # True
print("ABC".isupper())      # True
print("abc".islower())      # True
print("Hello World".istitle()) # True
print("   ".isspace())      # True
print("".isascii())         # True
print("①②③".isdecimal())   # False
print("123".isnumeric())    # True
```

9. How do you format strings using different methods?

python

```python
name = "Alice"
age = 25
score = 95.567

# f-strings (Python 3.6+)
message1 = f"Hello {name}, you are {age} years old"
message2 = f"Score: {score:.2f}%"

# format() method
message3 = "Hello {}, you are {} years old".format(name, age)
message4 = "Hello {name}, age: {age}".format(name=name, age=age)
message5 = "Score: {:.2f}%".format(score)

# % formatting (older method)
message6 = "Hello %s, you are %d years old" % (name, age)
message7 = "Score: %.2f%%" % score
```

10. How do you work with escape characters and raw strings?

python

```python
# Escape characters
text1 = "Line 1\nLine 2"           # Newline
text2 = "Tab\tSeparated"           # Tab
text3 = "He said \"Hello\""        # Quote
text4 = "Path: C:\\Users\\Name"     # Backslash

# Raw strings
path = r"C:\Users\Name\Documents"   # Raw string
regex = r"\d+\.\d+"                # Raw string for regex

# Triple quotes for multiline
multiline = """This is line 1
This is line 2
This is line 3"""
```

11. How do you reverse a string and check for palindromes?

python

```python
text = "racecar"

# Reverse using slicing
reversed_text = text[::-1]         # "racecar"

# Check palindrome
def is_palindrome(s):
    s = s.lower().replace(" ", "")
    return s == s[::-1]

print(is_palindrome("A man a plan a canal Panama"))  # True

# Reverse using join and reversed
reversed_text2 = "".join(reversed(text))
```

12. How do you work with string encoding and decoding?

python

```python
text = "Hello 世界"

# Encoding
utf8_bytes = text.encode('utf-8')     # b'Hello \xe4\xb8\x96\xe7\x95\x8c'
ascii_bytes = "Hello".encode('ascii') # b'Hello'

# Decoding
decoded_text = utf8_bytes.decode('utf-8')  # "Hello 世界"

# Handle encoding errors
try:
    "世界".encode('ascii')
except UnicodeEncodeError:
    safe_encode = "世界".encode('ascii', errors='ignore')  # b''
```

13. How do you use string translation and character mapping?

python

```python
text = "Hello World 123"

# Translation table
trans_table = str.maketrans("aeiou", "12345")
translated = text.translate(trans_table)  # "H2ll4 W4rld 123"

# Remove characters
remove_digits = str.maketrans("", "", "0123456789")
no_digits = text.translate(remove_digits)  # "Hello World "

# Character replacement mapping
char_map = str.maketrans({"H": "J", "W": "Universe"})
mapped = text.translate(char_map)  # "Jello Universeorld 123"
```

14. How do you perform advanced string operations?

python

```python
text = "The quick brown fox jumps over the lazy dog"

# Partition
before, sep, after = text.partition("fox")
# before: "The quick brown ", sep: "fox", after: " jumps over the lazy dog"

# Right partition
before2, sep2, after2 = text.rpartition("the")
# before2: "The quick brown fox jumps over ", sep2: "the", after2: " lazy dog"

# Zfill (zero padding)
number = "42"
padded = number.zfill(5)  # "00042"

# Center, ljust, rjust
centered = "Python".center(10, "*")  # "**Python**"
left_just = "Python".ljust(10, "-")  # "Python----"
right_just = "Python".rjust(10, "-") # "----Python"
```

15. How do you work with string templates and advanced formatting?

python

```python
from string import Template
import datetime

# String templates
template = Template("Hello $name, welcome to $place!")
result = template.substitute(name="Alice", place="Python World")

# Safe substitute (won't raise error for missing keys)
safe_result = template.safe_substitute(name="Bob")  # "Hello Bob, welcome to $place!"

# Advanced formatting
data = {"name": "Alice", "items": ["apple", "banana"], "total": 25.99}
formatted = "Customer {name} bought {items} for ${total:.2f}".format(**data)

# Date formatting
now = datetime.datetime.now()
date_str = f"Today is {now:%Y-%m-%d %H:%M:%S}"
```

Lists (Questions 16-30)

16. How do you create and access list elements?

python

```python
# Creating lists
numbers = [1, 2, 3, 4, 5]
mixed = [1, "hello", 3.14, True]
nested = [[1, 2], [3, 4], [5, 6]]
empty = []
from_range = list(range(5))        # [0, 1, 2, 3, 4]

# Accessing elements
first = numbers[0]                 # 1
last = numbers[-1]                 # 5
slice_result = numbers[1:4]        # [2, 3, 4]
nested_element = nested[0][1]      # 2
```

17. How do you modify lists (add, remove, update)?

python

```python
fruits = ["apple", "banana"]

# Adding elements
fruits.append("orange")            # ["apple", "banana", "orange"]
fruits.insert(1, "grape")          # ["apple", "grape", "banana", "orange"]
fruits.extend(["kiwi", "mango"])   # Add multiple elements
fruits += ["pear"]                 # Same as extend for single list

# Removing elements
removed = fruits.pop()             # Removes and returns last element
removed_index = fruits.pop(1)      # Removes and returns element at index 1
fruits.remove("banana")            # Removes first occurrence
del fruits[0]                      # Delete by index
fruits.clear()                     # Remove all elements
```

18. How do you search and count in lists?

python

```python
numbers = [1, 2, 3, 2, 4, 2, 5]

# Search
index = numbers.index(3)           # 2 (first occurrence)
try:
    index2 = numbers.index(2, 2)   # Search starting from index 2
except ValueError:
    print("Not found")

# Count
count = numbers.count(2)           # 3

# Membership
exists = 3 in numbers              # True
not_exists = 10 not in numbers     # True
```

19. How do you sort and reverse lists?

python

```python
numbers = [3, 1, 4, 1, 5, 9, 2]
names = ["Charlie", "Alice", "Bob"]

# Sort (modifies original)
numbers.sort()                     # [1, 1, 2, 3, 4, 5, 9]
numbers.sort(reverse=True)         # [9, 5, 4, 3, 2, 1, 1]
names.sort(key=len)                # Sort by length: ["Bob", "Alice", "Charlie"]

# Sorted (creates new list)
original = [3, 1, 4, 1, 5]
sorted_list = sorted(original)     # [1, 1, 3, 4, 5]
sorted_desc = sorted(original, reverse=True)  # [5, 4, 3, 1, 1]

# Reverse
numbers.reverse()                  # Reverses in place
reversed_list = list(reversed(numbers))  # Creates new reversed list
```

20. How do you use list comprehensions (basic to advanced)?

python

```python
# Basic list comprehension
squares = [x**2 for x in range(10)]           # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
evens = [x for x in range(20) if x % 2 == 0]  # [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

# With string operations
words = ["hello", "world", "python"]
upper_words = [word.upper() for word in words]  # ["HELLO", "WORLD", "PYTHON"]
lengths = [len(word) for word in words]         # [5, 5, 6]

# Nested list comprehension
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened = [num for row in matrix for num in row]  # [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Conditional expressions
result = [x if x > 0 else 0 for x in [-1, 2, -3, 4]]  # [0, 2, 0, 4]
```

21. How do you work with nested lists and matrices?

python

```python
# Creating matrices
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
zeros = [[0 for _ in range(3)] for _ in range(3)]

# Accessing matrix elements
element = matrix[1][2]              # 6
row = matrix[1]                     # [4, 5, 6]

# Matrix operations
def transpose(matrix):
    return [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]

transposed = transpose(matrix)      # [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

# Flattening nested lists
def flatten(lst):
    result = []
    for item in lst:
        if isinstance(item, list):
            result.extend(flatten(item))
        else:
            result.append(item)
    return result

nested = [1, [2, 3], [4, [5, 6]]]
flat = flatten(nested)              # [1, 2, 3, 4, 5, 6]
```

22. How do you copy lists (shallow vs deep)?

python

```python
import copy

original = [[1, 2], [3, 4]]

# Shallow copy
shallow1 = original.copy()
shallow2 = list(original)
shallow3 = original[:]

# Deep copy
deep = copy.deepcopy(original)

# Demonstrate the difference
original[0][0] = 99
print(shallow1)  # [[99, 2], [3, 4]] - affected by change
print(deep)      # [[1, 2], [3, 4]] - not affected
```

23. How do you use advanced list methods and operations?

python

```python
# List multiplication and addition
list1 = [1, 2, 3]
list2 = [4, 5, 6]
combined = list1 + list2           # [1, 2, 3, 4, 5, 6]
repeated = list1 * 3               # [1, 2, 3, 1, 2, 3, 1, 2, 3]

# Unpacking
a, b, c = [1, 2, 3]               # a=1, b=2, c=3
first, *middle, last = [1, 2, 3, 4, 5]  # first=1, middle=[2, 3, 4], last=5

# Zip with lists
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]
paired = list(zip(names, ages))    # [('Alice', 25), ('Bob', 30), ('Charlie', 35)]

# Enumerate
indexed = list(enumerate(names))   # [(0, 'Alice'), (1, 'Bob'), (2, 'Charlie')]
```

24. How do you filter and transform lists?

python

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Filter using list comprehension
evens = [x for x in numbers if x % 2 == 0]  # [2, 4, 6, 8, 10]

# Filter using filter() function
odds = list(filter(lambda x: x % 2 == 1, numbers))  # [1, 3, 5, 7, 9]

# Transform using map()
squared = list(map(lambda x: x**2, numbers))  # [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

# Multiple conditions
filtered = [x for x in numbers if x > 3 and x < 8]  # [4, 5, 6, 7]

# Filter with function
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

primes = [x for x in numbers if is_prime(x)]  # [2, 3, 5, 7]
```

25. How do you find min, max, and sum in lists?

python

```python
numbers = [3, 1, 4, 1, 5, 9, 2, 6]

# Basic aggregations
minimum = min(numbers)              # 1
maximum = max(numbers)              # 9
total = sum(numbers)                # 31
length = len(numbers)               # 8
average = sum(numbers) / len(numbers)  # 3.875

# Min/Max with key function
words = ["apple", "pie", "banana"]
shortest = min(words, key=len)      # "pie"
longest = max(words, key=len)       # "banana"

# Find indices of min/max
min_index = numbers.index(min(numbers))  # 1
max_index = numbers.index(max(numbers))  # 5

# Multiple lists
list1 = [1, 5, 3]
list2 = [2, 4, 6]
element_wise_max = [max(a, b) for a, b in zip(list1, list2)]  # [2, 5, 6]
```

26. How do you work with list slicing (advanced)?

```python
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# Basic slicing
first_five = numbers[:5]           # [0, 1, 2, 3, 4]
last_five = numbers[-5:]           # [5, 6, 7, 8, 9]
middle = numbers[3:7]              # [3, 4, 5, 6]

# Step slicing
evens = numbers[::2]               # [0, 2, 4, 6, 8]
odds = numbers[1::2]               # [1, 3, 5, 7, 9]
reversed_list = numbers[::-1]      # [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

# Negative indices and steps
last_three_reversed = numbers[-3:][::-1]  # [9, 8, 7]
every_third = numbers[::3]         # [0, 3, 6, 9]

# Slice assignment
numbers[2:5] = [20, 30, 40]       # Replace elements
numbers[1:1] = [10]               # Insert elements
del numbers[2:4]                  # Delete slice
```

27. How do you check list equality and comparison?

python

```python
list1 = [1, 2, 3]
list2 = [1, 2, 3]
list3 = [3, 2, 1]
list4 = [1, 2, 3, 4]

# Equality
print(list1 == list2)             # True
print(list1 == list3)             # False
print(list1 is list2)             # False (different objects)

# Element-wise comparison
print(list1 < list4)              # True (lexicographic comparison)
print([1, 2] < [1, 3])            # True

# Check if all/any elements meet condition
numbers = [2, 4, 6, 8]
print(all(x % 2 == 0 for x in numbers))  # True (all even)
print(any(x > 5 for x in numbers))       # True (at least one > 5)

# Set operations on lists
common = list(set(list1) & set(list3))   # [1, 2, 3] (intersection)
union = list(set(list1) | set(list3))    # [1, 2, 3] (union)
```

28. How do you partition and group lists?

python

```python
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Partition into chunks
def chunk_list(lst, chunk_size):
    return [lst[i:i + chunk_size] for i in range(0, len(lst), chunk_size)]

chunks = chunk_list(numbers, 3)    # [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10]]

# Group by condition
evens = [x for x in numbers if x % 2 == 0]
odds = [x for x in numbers if x % 2 == 1]

# Group by key function
from itertools import groupby
words = ["apple", "apricot", "banana", "blueberry", "cherry"]
grouped = {}
for key, group in groupby(sorted(words), key=lambda x: x[0]):
    grouped[key] = list(group)
# {'a': ['apple', 'apricot'], 'b': ['banana', 'blueberry'], 'c': ['cherry']}
```

29. How do you work with list performance and memory?

python

```python
import sys

# Memory usage
small_list = [1, 2, 3]
large_list = list(range(1000))
print(sys.getsizeof(small_list))   # Memory in bytes

# List vs Generator (memory efficient)
# List - stores all values in memory
list_comp = [x**2 for x in range(1000000)]

# Generator - computes values on demand
gen_comp = (x**2 for x in range(1000000))

# Efficient list operations
# Use extend() instead of multiple append() calls
efficient_list = []
efficient_list.extend(range(100))  # Better than loop with append

# Pre-allocate when size is known
known_size = [None] * 100
for i in range(100):
    known_size[i] = i**2
```

30. How do you use lists with advanced Python features?

python

```python
from collections import deque
from itertools import chain, combinations

# Using deque for efficient operations at both ends
dq = deque([1, 2, 3, 4, 5])
dq.appendleft(0)                    # [0, 1, 2, 3, 4, 5]
dq.popleft()                        # Remove from left

# Chain multiple lists
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list3 = [7, 8, 9]
chained = list(chain(list1, list2, list3))  # [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Generate combinations
items = [1, 2, 3]
combos = list(combinations(items, 2))  # [(1, 2), (1, 3), (2, 3)]

# List as stack
stack = []
stack.append(1)                     # Push
stack.append(2)
last = stack.pop()                  # Pop

# List as queue (not efficient, use deque instead)
queue = []
queue.append(1)                     # Enqueue
first = queue.pop(0)                # Dequeue (inefficient)
```

Tuples (Questions 31-40)

31. How do you create and access tuple elements?

python

```python
# Creating tuples
numbers = (1, 2, 3, 4, 5)
mixed = (1, "hello", 3.14, True)
single_element = (42,)              # Note the comma
empty_tuple = ()
without_parentheses = 1, 2, 3       # Also creates a tuple
from_list = tuple([1, 2, 3])

# Accessing elements
first = numbers[0]                  # 1
last = numbers[-1]                  # 5
slice_result = numbers[1:4]         # (2, 3, 4)
length = len(numbers)               # 5
```

32. How do you work with tuple immutability and operations?

python

```python
coordinates = (3, 4)

# Tuples are immutable - these would raise errors:
# coordinates[0] = 5  # TypeError
# coordinates.append(5)  # AttributeError

# But you can create new tuples
new_coordinates = coordinates + (5,)  # (3, 4, 5)
repeated = coordinates * 3            # (3, 4, 3, 4, 3, 4)

# Membership and comparison
print(3 in coordinates)               # True
print(coordinates > (2, 3))           # True (lexicographic comparison)

# Nested tuples with mutable objects
nested = ([1, 2], [3, 4])
nested[0].append(3)                   # This works! - ([1, 2, 3], [3, 4])
# nested[0] = [1, 2, 3]              # This doesn't work!
```

33. How do you unpack tuples and use them in functions?

python

```python
point = (3, 4)
person = ("Alice", 25, "Engineer")

# Basic unpacking
x, y = point                        # x=3, y=4
name, age, job = person             # name="Alice", age=25, job="Engineer"

# Extended unpacking
numbers = (1, 2, 3, 4, 5)
first, *middle, last = numbers      # first=1, middle=[2, 3, 4], last=5
first_two, *rest = numbers          # first_two=1, rest=[2, 3, 4, 5]

# Swapping variables
a, b = 10, 20
a, b = b, a                         # Swap using tuple unpacking

# Function returning multiple values
def get_name_age():
    return "Bob", 30

name, age = get_name_age()

# Function with tuple parameter
def calculate_distance(point1, point2):
    x1, y1 = point1
    x2, y2 = point2
    return ((x2-x1)**2 + (y2-y1)**2)**0.5

distance = calculate_distance((0, 0), (3, 4))  # 5.0
```

34. How do you use tuple methods and operations?

```python
numbers = (1, 2, 3, 2, 4, 2, 5)

# Count occurrences
count_2 = numbers.count(2)          # 3
count_1 = numbers.count(1)          # 1

# Find index
index_3 = numbers.index(3)          # 2
try:
    index_2_after_3 = numbers.index(2, 3)  # 3 (search starting from index 3)
except ValueError:
    print("Not found")

# Conversion
list_version = list(numbers)        # [1, 2, 3, 2, 4, 2, 5]
back_to_tuple = tuple(list_version) # (1, 2, 3, 2, 4, 2, 5)

# Iteration
for i, value in enumerate(numbers):
    print(f"Index {i}: {value}")
```

35. How do you use tuples as dictionary keys and in sets?

python

```python
# Tuples as dictionary keys (hashable)
coordinates_dict = {
    (0, 0): "origin",
    (1, 0): "right",
    (0, 1): "up",
    (1, 1): "diagonal"
}

location = coordinates_dict[(1, 1)]  # "diagonal"

# Tuples in sets
point_set = {(0, 0), (1, 1), (2, 2)}
point_set.add((3, 3))
unique_points = point_set            # {(0, 0), (1, 1), (2, 2), (3, 3)}

# Cannot use lists as keys (not hashable)
# bad_dict = {[1, 2]: "value"}      # TypeError

# Complex tuple keys
student_grades = {
    ("Alice", "Math"): 95,
    ("Alice", "Science"): 87,
    ("Bob", "Math"): 82,
    ("Bob", "Science"): 90
}

alice_math = student_grades[("Alice", "Math")]  # 95
```

36. How do you work with named tuples?

python

```python
from collections import namedtuple

# Define named tuple
Point = namedtuple('Point', ['x', 'y'])
Person = namedtuple('Person', 'name age city')  # Can use string with spaces

# Create instances
p1 = Point(3, 4)
p2 = Point(x=1, y=2)
person1 = Person("Alice", 25, "New York")

# Access by name or index
print(p1.x)                         # 3
print(p1[0])                        # 3 (same as above)
print(person1.name)                 # "Alice"

# Named tuple methods
print(p1._fields)                   # ('x', 'y')
p3 = p1._replace(x=5)               # Point(x=5, y=4)
point_dict = p1._asdict()           # {'x': 3, 'y': 4}

# Create from iterable
coords = [10, 20]
p4 = Point._make(coords)            # Point(x=10, y=20)
```

37. How do you sort and compare tuples?

python

```python
# List of tuples
students = [
    ("Alice", 85, 22),
    ("Bob", 90, 25),
    ("Charlie", 78, 23),
    ("Diana", 92, 21)
]

# Sort by first element (default)
sorted_by_name = sorted(students)

# Sort by specific element
sorted_by_grade = sorted(students, key=lambda x: x[1], reverse=True)
sorted_by_age = sorted(students, key=lambda x: x[2])

# Multiple
```