Python Coding Essentials

1. Data Types & Basic Operations

Numbers

```
# Integer and float operations
x = 10
y = 3.14
print(x // 3)  # Floor division: 3
print(x % 3)   # Modulo: 1
print(2 ** 3)  # Exponentiation: 8
```

Strings

```
# String manipulation
s = "Hello World"
print(s[0])        # 'H'
print(s[-1])       # 'd'
print(s[0:5])      # 'Hello'
print(s.split())   # ['Hello', 'World']
print(' '.join(['a', 'b', 'c']))  # 'a b c'

# String methods
print(s.lower())   # 'hello world'
print(s.replace('World', 'Python'))  # 'Hello Python'
```

2. Lists - Most Important for Interviews

Basic Operations

```
# Creating and accessing
arr = [1, 2, 3, 4, 5]
print(arr[0])      # 1
print(arr[-1])     # 5
print(arr[1:4])    # [2, 3, 4]

# Adding elements
arr.append(6)      # [1, 2, 3, 4, 5, 6]
arr.insert(0, 0)   # [0, 1, 2, 3, 4, 5, 6]
arr.extend([7, 8]) # [0, 1, 2, 3, 4, 5, 6, 7, 8]

# Removing elements
arr.pop()          # Removes last element
arr.pop(0)         # Removes first element
arr.remove(3)      # Removes first occurrence of 3
```

List Comprehensions (Very Important!)

```
# Basic syntax: [expression for item in iterable if condition]
nums = [1, 2, 3, 4, 5]
squares = [x**2 for x in nums]  # [1, 4, 9, 16, 25]
evens = [x for x in nums if x % 2 == 0]  # [2, 4]

# Nested comprehensions
matrix = [[1, 2], [3, 4], [5, 6]]
flattened = [item for row in matrix for item in row]  # [1, 2, 3, 4, 5, 6]
```

3. Dictionaries - Critical for Many Problems

```
# Creating and accessing
d = {'a': 1, 'b': 2, 'c': 3}
print(d['a'])      # 1
print(d.get('d', 0))  # 0 (default value)

# Adding/updating
d['d'] = 4
d.update({'e': 5, 'f': 6})

# Useful methods
print(d.keys())    # dict_keys(['a', 'b', 'c', 'd', 'e', 'f'])
print(d.values())  # dict_values([1, 2, 3, 4, 5, 6])
print(d.items())   # dict_items([('a', 1), ('b', 2), ...])

# Dictionary comprehension
squares_dict = {x: x**2 for x in range(5)}  # {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

4. Sets - Great for Duplicates and Lookups

```
# Creating sets
s1 = {1, 2, 3, 4}
s2 = set([3, 4, 5, 6])

# Set operations
print(s1 & s2)     # Intersection: {3, 4}
print(s1 | s2)     # Union: {1, 2, 3, 4, 5, 6}
print(s1 - s2)     # Difference: {1, 2}

# Adding/removing
s1.add(5)
s1.discard(1)      # Won't error if element doesn't exist
```

5. Control Flow

Loops

```
# For loops
for i in range(5):        # 0, 1, 2, 3, 4
    print(i)

for i, val in enumerate(['a', 'b', 'c']):
    print(i, val)         # 0 a, 1 b, 2 c

# While loops
i = 0
while i < 5:
    i += 1

# Loop control
for i in range(10):
    if i == 3:
        continue          # Skip this iteration
    if i == 7:
        break             # Exit loop
    print(i)
```

Conditionals

```
x = 10
if x > 5:
    print("Greater than 5")
elif x == 5:
    print("Equal to 5")
else:
    print("Less than 5")

# Ternary operator
result = "positive" if x > 0 else "negative"
```

6. Functions

```python
def fibonacci(n):
    """Calculate nth Fibonacci number"""
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

# Lambda functions
square = lambda x: x**2
numbers = [1, 2, 3, 4, 5]
squared = list(map(square, numbers))  # [1, 4, 9, 16, 25]

# Filter and map
evens = list(filter(lambda x: x % 2 == 0, numbers))  # [2, 4]
```

7. Common Built-in Functions for Interviews

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

# Sorting
print(sorted(numbers))           # [1, 1, 2, 3, 4, 5, 6, 9]
print(sorted(numbers, reverse=True))  # [9, 6, 5, 4, 3, 2, 1, 1]

# Min, max, sum
print(min(numbers))    # 1
print(max(numbers))    # 9
print(sum(numbers))    # 31

# Length and all/any
print(len(numbers))    # 8
print(all([True, True, False]))  # False
print(any([False, False, True])) # True

# Zip
names = ['Alice', 'Bob', 'Charlie']
ages = [25, 30, 35]
for name, age in zip(names, ages):
    print(f"{name} is {age} years old")
```

8. String Formatting

```
name = "Alice"
age = 25

# f-strings (preferred in Python 3.6+)
print(f"My name is {name} and I'm {age} years old")

# format method
print("My name is {} and I'm {} years old".format(name, age))
```

9. Error Handling

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    print("This always executes")
```

10. Common Interview Patterns

Two Pointers

```python
def two_sum_sorted(arr, target):
    """Find two numbers that sum to target in sorted array"""
    left, right = 0, len(arr) - 1
    while left < right:
        current_sum = arr[left] + arr[right]
        if current_sum == target:
            return [left, right]
        elif current_sum < target:
            left += 1
        else:
            right -= 1
    return []
```

Sliding Window

```python
def max_sum_subarray(arr, k):
    """Find maximum sum of subarray of size k"""
    if len(arr) < k:
        return -1

    # Calculate sum of first window
    window_sum = sum(arr[:k])
    max_sum = window_sum

    # Slide the window
    for i in range(len(arr) - k):
        window_sum = window_sum - arr[i] + arr[i + k]
        max_sum = max(max_sum, window_sum)

    return max_sum
```

Hash Map for Counting

```python
def char_count(s):
    """Count frequency of each character"""
    count = {}
    for char in s:
        count[char] = count.get(char, 0) + 1
    return count

# Or using Counter from collections
from collections import Counter
count = Counter("hello")  # Counter({'l': 2, 'h': 1, 'e': 1, 'o': 1})
```

Quick Tips for Interviews

1. **Time Complexity**: Always discuss Big O notation
   - O(1): Constant time
   - O(log n): Logarithmic (binary search)
   - O(n): Linear (single loop)
   - O(n log n): Linearithmic (efficient sorting)
   - O(n^2): Quadratic (nested loops)
2. **Space Complexity**: Consider memory usage
   - In-place algorithms use O(1) extra space
   - Recursive solutions use O(h) space where h is recursion depth
3. **Edge Cases**: Always consider:
   - Empty inputs
   - Single element
   - Duplicates
   - Negative numbers
   - Very large inputs
4. **Python-Specific Advantages**:
   - List slicing: `arr[start:end:step]`
   - Multiple assignment: `a, b = b, a`
   - `in` operator for membership testing
   - Built-in `sorted()`, `min()`, `max()`, `sum()`

Practice these concepts and you'll be well-prepared for most Python interview questions!