Python Master Study Guide: Week 1

Day 1 – Setting up Python & Basic Interaction

Focus on the environment and the bridge between the user and the program.

`print()` and `input()`

- **Explanation:** `print()` sends data to the console. `input()` pauses the program to collect text from the user. Note that `input()` always returns a **string**.
- **Variables:** Containers for storing data values. Python is dynamically typed, so you don't need to declare types.

```
# Simple Input and Output
user_name = input("Enter your name: ")
age = 25  # Variable assignment
print(f"Hello {user_name}, you are {age} years old.")
```

Day 2 – Python Basics Continued

Understanding how Python interprets different kinds of information.

Data Types & Operators

- **Types:** `int` (10), `float` (10.5), `str` ("Hi"), `bool` (True/False).
- **Conditionals:** `if-elif-else` blocks control the flow based on boolean logic.

```
# Data Types and Conditionals
price = 19.99       # float
quantity = 5        # int
is_available = True # bool

if is_available and (price * quantity > 50):
    print("Bulk discount applied!")
elif not is_available:
    print("Out of stock.")
else:
    print("Standard pricing.")
```

Day 3 – Control Flow

Repeating logic efficiently.

Loops and Identifiers

- **Loops:** `for` is used for iterating over a sequence; `while` runs as long as a condition is true.
- **Identifiers:** Names given to entities (variables, functions). Must start with a letter or `_`.

```
# Iteration
for i in range(3):
    print(f"Iteration {i}")

count = 0
while count < 2:
    print("Looping...")
    count += 1
```

Day 4 – Python Functions (Part 1)

The foundation of DRY (Don't Repeat Yourself) programming.

`def`, `return`, and Scope

- `pass`: A null statement used as a placeholder.
- **Scope:** Variables inside a function are **local**; variables outside are **global**.

```
global_var = "I am global"

def calculate_area(radius):
    if radius < 0:
        pass # Placeholder for logic to be added later
    pi = 3.14  # Local variable
    return pi * (radius ** 2)

print(calculate_area(5))
```

Day 5 – Python Functions (Part 2)

Handling dynamic arguments and functional programming.

Advanced Functions

- `*args` / `**kwargs`: `*args` collects extra positional arguments as a tuple; `**kwargs` collects keyword arguments as a dictionary.
- `lambda`: Anonymous one-line functions.
- **Decorators:** Functions that modify the behavior of another function.

```
# Lambda & Map
numbers = [1, 2, 3]
squared = list(map(lambda x: x**2, numbers))

# Args and Kwargs
```

```
def flexible_func(*args, **kwargs):
    print(args)   # (1, 2)
    print(kwargs) # {'key': 'value'}

flexible_func(1, 2, key="value")
```

Day 6 – Built-In Data Structures (Part 1)

Organizing collections of data. Here are the exhaustive common methods:

1. Lists `[]` (Mutable)

Method	Description
`append(x)`	Adds item `x` to the end
`extend(iterable)`	Adds all elements of an iterable to the end
`insert(i, x)`	Inserts item `x` at index `i`
`remove(x)`	Removes the first occurrence of item `x`
`pop([i])`	Removes and returns item at index `i` (default last)
`clear()`	Removes all items
`index(x)`	Returns index of first occurrence of `x`
`count(x)`	Returns number of times `x` appears
`sort()`	Sorts the list in place
`reverse()`	Reverses the list in place
`copy()`	Returns a shallow copy

2. Dictionaries `{}` (Key-Value Pairs)

Method	Description
`clear()`	Removes all elements
`copy()`	Returns a copy
`get(key, default)`	Returns value for key, or default if not found

`items()`	Returns list of (key, value) tuples
`keys()`	Returns list of keys
`values()`	Returns list of values
`pop(key)`	Removes and returns value for specific key
`popitem()`	Removes and returns last inserted (key, value) pair
`update(dict2)`	Updates dictionary with `dict2`

3. Strings `""` (Immutable)

Method	Description
`upper()` / `lower()`	Converts case
`strip()`	Removes leading/trailing whitespace
`replace(old, new)`	Replaces substring
`split(sep)`	Splits string into a list
`join(iterable)`	Joins elements of iterable with the string as separator
`find(sub)`	Returns index of first occurrence of `sub`
`startswith(prefix)`	Returns True if starts with prefix

Day 7 – Built-In Data Structures (Part 2)

Sets `{}` (Unique Elements)

Method	Description
`add(x)`	Adds element `x`
`remove(x)` / `discard(x)`	Removes `x` (`discard` doesn't error if missing)
`pop()`	Removes and returns an arbitrary element
`union(other)`	Returns a set containing all elements from both
`intersection(other)`	Returns only elements found in both

`difference(other)`	Returns elements in this set but not the other

List Comprehensions

A concise way to create lists using existing iterables. **Syntax:** `[expression for item in iterable if condition]`

```
# Create a list of squares for even numbers only
evens_squared = [x**2 for x in range(10) if x % 2 == 0]
```