Python Interview Study Guide - Day One

1. Key Features of Python:

Python is a high-level, interpreted programming language with several powerful features:

- Easy to learn: Simple syntax, similar to English.
- Dynamically typed: You don't need to declare variable types explicitly.
- Object-Oriented: Supports classes and objects.
- Rich Libraries: Includes modules for math, OS interaction, networking, web development, etc.
- Community Support: Huge developer community and open-source packages.

2. Python 2 vs Python 3:

Python 3 is the present and future of Python development. Key differences include:

- Print: Python 2: print "Hello", Python 3: print("Hello")
- Integer Division: Python 2: 3/2 = 1, Python 3: 3/2 = 1.5
- Unicode: Python 3 uses Unicode strings by default.
- Library Support: Most modern libraries support Python 3 only.

3. Python Data Types:

Python provides several built-in data types:

- Numeric Types: int, float, complex
- Text Type: str
- Boolean Type: bool (True or False)
- Sequence Types: list (mutable), tuple (immutable), range
- Mapping Type: dict (key-value pairs)
- Set Types: set, frozenset

Example:

```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
my_dict = {"name": "Alice", "age": 30}
my_set = {1, 2, 3}
```

4. How to Create a Function in Python:

Functions are defined using the 'def' keyword.

Example:

```
def sum_numbers(a, b):
    return a + b

result = sum_numbers(5, 3)
print(result)  # Output: 8
```

5. Difference Between Tuple and List:

- Tuple: Immutable and uses parentheses ()
- List: Mutable and uses square brackets []

Example:

```
my_tuple = (1, 2, 3)
my_list = [1, 2, 3]
# my_tuple[0] = 10  # This would raise an error
```

6. List Comprehensions:

List comprehensions offer a concise way to create lists.

Example:

```
students = {"student_1": "Erico", "student_2": "John"}
student_names = [name for key, name in students.items()]
print(student_names)  # Output: ['Erico', 'John']
```

7. Difference Between '==' and 'is':

- '==' checks value equality.
- 'is' checks object identity (whether both point to the same memory location).

Example:

```
a = [1, 2]
```

```
b = [1, 2]
print(a == b)  # True (same content)
print(a is b)  # False (different objects)
```

8. Lambda Functions:

Lambda functions are anonymous functions defined in a single line.

Example:

```
add = lambda x, y: x + y
print(add(5, 4))  # Output: 9
```

9. Exception Handling:

Python uses try-except blocks for error handling. You can also use else and finally.

Example:

```
def get_second_value(test_list):
    try:
        second_value = test_list[1]
    except IndexError:
        print("Second item not found.")
    except TypeError:
        print(f"Invalid type: {type(test_list)}")
    else:
        print(f"Second value: {second_value}")
    finally:
        print("Operation complete.")

get_second_value([10, 20])
```

10. Decorators:

Decorators allow you to wrap one function around another.

Example:

```
def my_decorator(func):
    def wrapper():
        print("Starting function...")
        func()
        print("Function ended.")
    return wrapper

@my_decorator
def say_hello():
    print("Hello!")

say_hello()
```