Python Functions and Data Structures: Student Handout

Introduction to Functions in Python

What is a Function?

A **function** is a block of code designed to perform a specific task. It allows you to write code once and reuse it multiple times.

Defining and Calling Functions

Defining a Function

Use the `def` keyword to define a function. Here's the syntax:

```
Python
def function_name():
    # Code block that performs a task
    print("Hello, this is a function!")
```

Calling a Function

Invoke a function by writing its name followed by parentheses:

```
Python
def greet():
    print("Hello, welcome to the Python class.")

greet()  # Output: Hello, welcome to the Python class.
```

Understanding Function Arguments and Return Values

Function Arguments

Functions can take inputs, known as **arguments**.

```python
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")  # Output: Hello, Alice!
```

Return Values

Functions can return results using the `return` keyword.

```python
def add_numbers(a, b):
    return a + b

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

Scope of Variables: Local and Global

Local Variables

Variables defined inside a function are **local** and cannot be accessed outside.

```python
def my_function():
    x = 10
    print(x)

my_function()  # Output: 10
# print(x)  # Error: x is not defined outside the function
```

Global Variables

Variables defined outside any function are **global** and can be accessed anywhere.

```python
x = 20

def my_function():
    print(x)
```

```
my_function()  # Output: 20
print(x)  # Output: 20
```

Writing Simple Functions to Automate Tasks

Automate repetitive tasks by defining functions.

```python
def calculate_area(length, width):
    return length * width

area1 = calculate_area(5, 3)
area2 = calculate_area(7, 2)

print(area1)  # Output: 15
print(area2)  # Output: 14
```

Overview of Python Data Structures

1. Lists

A **list** is an ordered, changeable collection.

```python
my_list = [1, 2, 3, 4]
my_list.append(5)
print(my_list)  # Output: [1, 2, 3, 4, 5]
```

2. Tuples

A **tuple** is an ordered, immutable collection.

```python
my_tuple = (1, 2, 3)
print(my_tuple)  # Output: (1, 2, 3)
```

3. Dictionaries

A **dictionary** is a collection of key-value pairs.

```python
my_dict = {"name": "Alice", "age": 25}
print(my_dict["name"])  # Output: Alice
```

4. Sets

A **set** is a collection of unique items.

```python
my_set = {1, 2, 3, 3}
print(my_set)  # Output: {1, 2, 3}
```

Performing Operations on Data Structures

- **Lists**: Add items with `append()`, remove with `remove()`, access by index.
- **Dictionaries**: Add/update key-value pairs, access values by keys.
- **Sets**: Add items with `add()`, remove with `remove()`, perform set operations like union and intersection.

Activity: Write a Python Script Using Lists and Dictionaries

Task: Store names and ages in a dictionary and print personalized greetings.

```python
# Step 1: Create a dictionary with names and ages
people = {
    "Alice": 25,
    "Bob": 30,
    "Charlie": 22
}

# Step 2: Define a function to greet each person
def greet_person(name, age):
    print(f"Hello, {name}! You are {age} years old.")

# Step 3: Call the function for each person in the dictionary
for name, age in people.items():
    greet_person(name, age)
```

Output:

```
Unset
Hello, Alice! You are 25 years old.
Hello, Bob! You are 30 years old.
Hello, Charlie! You are 22 years old.
```

Conclusion

In this session, we covered the basics of **functions** in Python, including defining and calling them, working with **arguments** and **return values**, and understanding the scope of variables. We also explored Python's core **data structures**—lists, tuples, dictionaries, and sets—and learned how to perform operations on them. Functions help make your code organized, reusable, and efficient. Keep practicing to enhance your programming skills!