رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Python Programming

Session 1: (Data Structures and Control Flow)

Agenda:

1	Data Containers
2	Control Flow

Data Containers

Data Containers

Python provides several built-in data structures (or containers) to organize and manage data efficiently. These include lists, tuples, sets, and dictionaries.

Lists

A list is an ordered, mutable collection of items. Lists can contain items of different types, including other lists.

Syntax

Lists are defined using square brackets [ ]

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

Key Characteristics of list

- **Ordered**: Lists allow access to items using an index.
- **Mutable**: You can add, delete, or modify items in a list.
- **Non-Unique Items**: Lists can contain duplicate elements.
- **Mixed Data Types**: Lists can hold items of different data types.

Data Containers

Lists Operations

Indexing

Access elements using their index (starting at 0)

```
print(fruits[0])  # Output: apple
```

```
apple
```

Slicing

Access a range of elements

```
print(fruits[0:2])  # Output: ['apple', 'banana']
```

```
['apple', 'banana']
```

Data Containers

Lists Operations

Adding Elements

Use append() to add an item, insert() to add at a specific index.

```
fruits.append("orange")
fruits.insert(1, "mango")
```

Removing Elements

Use remove() to delete an item, pop() to remove by index.

```
fruits.remove("banana")
fruits.pop(2)
```

```
'cherry'
```

Tuples

A tuple is an ordered, immutable collection of items. Once defined, its contents cannot be changed.

Syntax

Tuples are defined using parentheses ( )

```
coordinates = (10, 20)
```

Operations

Indexing and Slicing: Similar to lists.

```
print(coordinates[0])  # Output: 10
```

```
10
```

Immutability

Cannot change elements, which makes tuples faster for read-only data.

Tuples

Key Characteristics of Tuples

- **Ordered**: Tuples allow access to items using an index.
- **Immutable**: You cannot add, delete, or modify items in a tuple.
- **Non-Unique Items**: Tuples can contain duplicate elements.
- **Mixed Data Types**: Tuples can hold items of different data types.

Sets

A set is an unordered collection of unique items. Sets are used for membership testing and eliminating duplicate entries.

Syntax

Sets are defined using curly braces {} or the set() function

```
unique_numbers = {1, 2, 3, 4}
```

Operations

Adding Elements: Use add()

```
unique_numbers.add(5)
```

Sets

Key Characteristics of Sets

- **Unordered**: Sets do not maintain a specific order of items.
- **No Indexing/Slicing**: Sets do not support indexing or slicing operations.
- **Immutable Data Types**: Sets can only contain immutable data types (e.g., numbers, strings, tuples).
- **Unique Items**: Sets automatically remove duplicate items, ensuring all elements are unique.

Sets

Operations

Removing Elements: Use remove() or discard()

```
unique_numbers.remove(2)
```

Set Operations: Union, intersection, difference.

```
set_a = {1, 2, 3}
set_b = {3, 4, 5}
print(set_a | set_b)  # Union: {1, 2, 3, 4, 5}
print(set_a & set_b)  # Intersection: {3}
print(set_a - set_b)  # Difference: {1, 2}
```

```
{1, 2, 3, 4, 5}
{3}
{1, 2}
```

Dictionaries

A dictionary is an unordered collection of key-value pairs. Keys must be unique and immutable (e.g., strings, numbers).

Syntax

Defined using curly braces { } with key-value pairs separated by colons.

```
student = {"name": "Alice", "age": 20, "courses": ["Math", "Science"]}
```

Operations

Accessing Values: Use keys.

```
print(student["name"])  # Output: Alice
```

```
Alice
```

Adding/Updating: Assign a value to a key.

```
[15] student["age"] = 21
     student["grade"] = "A"
```

Dictionaries

Operations

Removing Items: Use pop() to remove a key-value pair.

```
student.pop("age")
```

```
21
```

Key Characteristics of Dictionaries

- **Key-Value Pairs**: Dictionaries store data as key-value pairs.
- **Immutable Keys**: Dictionary keys must be immutable (e.g., numbers, strings, tuples).
- **Flexible Values**: Dictionary values can be of any data type.
- **Unique Keys**: Each key in a dictionary must be unique.
- **Unordered**: Dictionaries are not ordered; elements are accessed using keys.

Comparing Lists, Tuples, Sets, Dictionaries

List	Tuple	Set	Dictionary
[] or list()	() or tuple()	{} or set()	{} or dict()
Ordered	Ordered	Unordered	Unordered
Changeable	Unchangeable	Unchangeable	Changeable
Indexed	Indexed	Unindexed	Key Value pair
Allows Duplicates	Allows Duplicates	No Duplicates	No Duplicates
Allows Slicing	Allows Slicing	No Slicing	No Slicing

Control Flow

Control Flow

Control flow statements are essential in programming as they enable the execution of specific blocks of code based on conditions. This allows for dynamic decision-making within a program. In Python, the primary control flow statements include if, elif, and else. We'll cover these concepts with detailed explanations and multiple examples to illustrate their use.

Control Flow

If Statements

The if statement is used to test a specific condition. If the condition evaluates to True, the code block under the if statement is executed.

Syntax

```
if condition:
    # Code to execute if condition is true
```

If Statements

Examples

Check Positive Number

```
number = 5
if number > 0:
    print("The number is positive.")
```

```
The number is positive.
```

This code checks if the number is greater than zero. Since 5 > 0 is true, it prints "The number is positive.

If Statements

Examples

Check String Length

```
name = "Alice"
if len(name) > 3:
    print("The name is longer than 3 characters.")
```

```
The name is longer than 3 characters.
```

The code checks if the length of the string name is greater than 3. Since "Alice" has 5 characters, the condition is true, and it prints "The name is longer than 3 characters."

If Statements

Examples

Check Membership in List

```
fruits = ["apple", "banana", "cherry"]
if "banana" in fruits:
    print("Banana is in the list.")
```

```
Banana is in the list.
```

The code checks if "banana" is an item in the fruits list. Since it is, the condition is true, and it prints "Banana is in the list."

If Statements

Examples

Check if Variable is Not None

```
result = None
if result is None:
    print("Result is not set yet.")
```

```
Result is not set yet.
```

This code checks if result is None. Since it is, the condition is true, and it prints "Result is not set yet."

Control Flow

If-Else Statements

The if-else statement provides an alternative block of code that is executed if the condition is False.

If-Else Statements

Examples

Check Even or Odd Number

```
number = 4
if number % 2 == 0:
    print("Even number")
else:
    print("Odd number")
```

```
Even number
```

This code checks if number is even by using the modulus operator %. Since 4 % 2 == 0 is true, it prints "Even number."

If-Else Statements

Examples

Age Verification

```
age = 17
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

```
You are not eligible to vote.
```

The code checks if age is 18 or older. Since age is 17, the condition is false, so it prints "You are not eligible to vote."

If-Else Statements

Examples

Check for Empty List

```
items = []
if items:
    print("The list is not empty.")
else:
    print("The list is empty.")
```

```
The list is empty.
```

The code checks if items is a non-empty list. Since items is empty, the condition is false, and it prints "The list is empty."

If-Else Statements

Examples

Temperature Check

```
temperature = 15
if temperature > 20:
    print("It's warm outside.")
else:
    print("It's cold outside.")
```

```
It's cold outside.
```

This code checks if the temperature is greater than 20. Since 15 is not greater than 20, it prints "It's cold outside."

Control Flow

If-Elif-Else Statements

The if-elif-else statement allows checking multiple conditions sequentially. Only the block of the first True condition is executed. If none of the conditions are True, the else block is executed.

If-Elif-Else Statements

Examples

Grading System

```
score = 72
if score >= 90:
    print("Grade A")
elif score >= 80:
    print("Grade B")
elif score >= 70:
    print("Grade C")
else:
    print("Grade D")
```

```
Grade C
```

This code assigns grades based on score. Since the score is 72, it matches the elif score >= 70: condition, so it prints "Grade C."

If-Elif-Else Statements

Examples

Traffic Light Signal

```
traffic_light = "yellow"
if traffic_light == "red":
    print("Stop")
elif traffic_light == "yellow":
    print("Slow down")
elif traffic_light == "green":
    print("Go")
else:
    print("Invalid signal")
```

```
Slow down
```

This code checks the color of traffic_light. Since it is "yellow," it matches the second condition and prints "Slow down."

If-Elif-Else Statements

Examples

Classify Age Group

```
age = 35
if age < 13:
    print("Child")
elif age < 18:
    print("Teenager")
elif age < 65:
    print("Adult")
else:
    print("Senior")
```

```
Adult
```

The code classifies age into different groups. Since 35 is less than 65 but more than 18, it matches the "Adult" category and prints "Adult."

If-Elif-Else Statements

Examples

Check Temperature Range

```
temperature = 30
if temperature < 0:
    print("Freezing")
elif temperature <= 20:
    print("Cold")
elif temperature <= 30:
    print("Warm")
else:
    print("Hot")
```

```
Warm
```

The code checks which temperature range temperature falls into. Since it is 30, it matches the elif temperature <= 30: condition and prints "Warm."

Control Flow

Nested If Statements

If statements can be nested within other if statements to create complex conditions.

Nested If Statements

Examples

Permission Check with Age

```
age = 17
has_permission = True
if age < 18:
    if has_permission:
        print("You can enter with permission.")
    else:
        print("You cannot enter.")
else:
    print("You are allowed to enter.")
```

```
You can enter with permission.
```

This code checks if age is less than 18 and then checks if has_permission is True. Since both conditions are met, it prints "You can enter with permission."

Nested If Statements

Examples

Nested Condition for Shopping

```
has_discount = True
is_member = False
if has_discount:
    if is_member:
        print("You get a 20% discount.")
    else:
        print("You get a 10% discount.")
else:
    print("No discount available.")
```

```
You get a 10% discount.
```

This code first checks if has_discount is True and then checks if is_member is also True. Since is_member is False, it prints "You get a 10% discount."

Nested If Statements

Examples

Nested Temperature and Time Check

```
temperature = 22
time_of_day = "morning"
if temperature > 20:
    if time_of_day == "morning":
        print("It's warm and morning time.")
    else:
        print("It's warm but not morning.")
else:
    print("It's cold.")
```

```
It's warm and morning time.
```

This code checks if temperature is greater than 20 and then checks if it's morning. Since both conditions are true, it prints "It's warm and morning time."

Nested If Statements

Examples

Bank Account Balance Check

```
balance = 500
withdraw_amount = 300
if balance >= withdraw_amount:
    if withdraw_amount > 0:
        print("Transaction successful.")
    else:
        print("Invalid amount to withdraw.")
else:
    print("Insufficient funds.")
```

```
Transaction successful.
```

This code checks if the balance is sufficient for the withdraw_amount and then checks if the amount is positive. Since both conditions are true, it prints "Transaction successful."