Data Structures and Algorithms
(ES221)

A Quick Talk About Computer Programming (2)

Dr. Zubair Ahmad

Basic Control Structures

- A sequence is a series of statements that execute one after another
- A selection(branch) statement is used to determine which of two different statements to execute depending on certain conditions
- A looping(repetition) statement is used to repeat statements while certain conditions are met
- A subprogram is a smaller part of another program; a collection of subprograms solves the original problem

SEQUENCE

 # Example: Sequence of Statements in Python

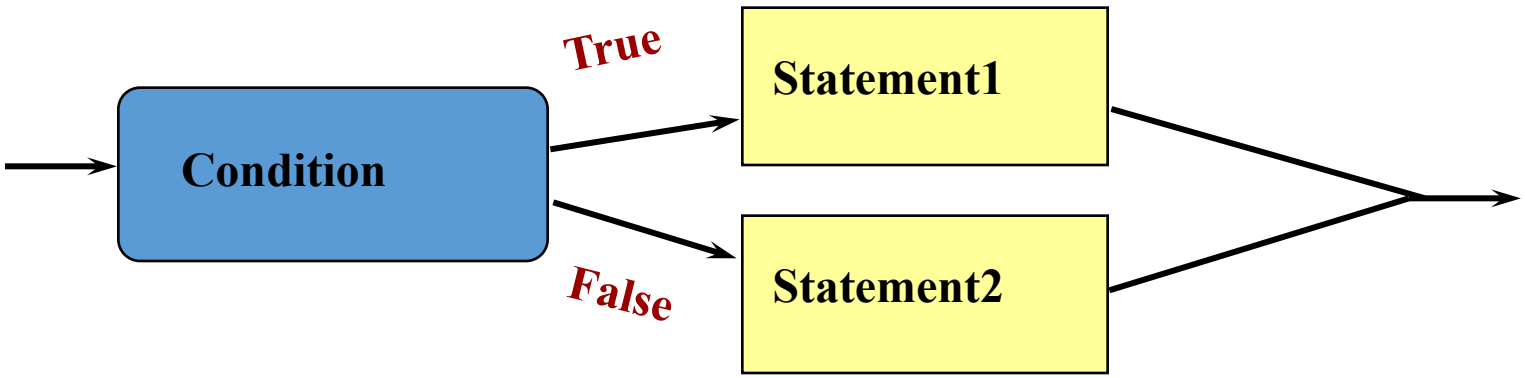
```
# Statement 1: Assign a value to a variable
x = 10

# Statement 2: Perform a calculation
y = x * 2

# Statement 3: Print the result
print("The value of y is:", y)

# Statement 4: Conditional check
if y > 15:
    print("y is greater than 15")
else:
    print("y is 15 or less")
```

SELECTION (Branch)



SELECTION (Branch)

Scenario: Online Shopping Discount
Imagine you're running an online store that offers discounts based on the customer's total purchase amount:

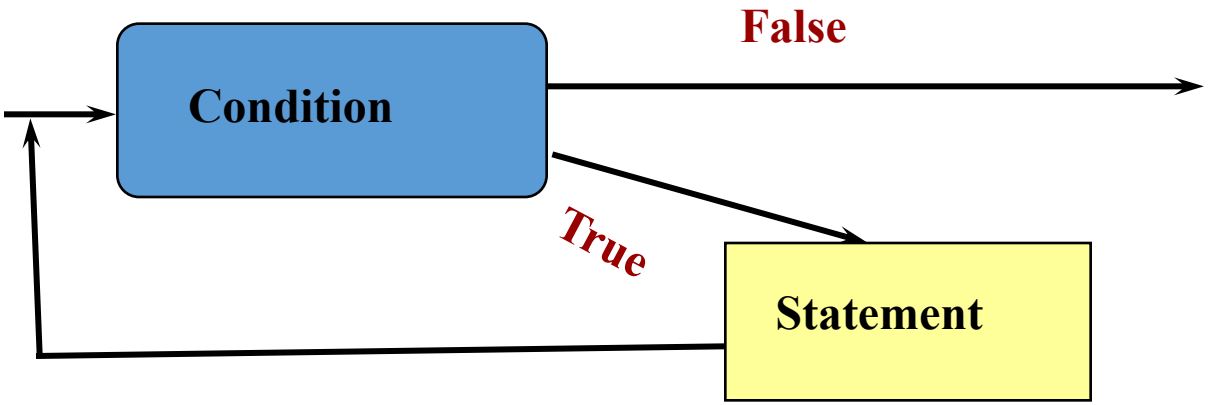
1.If the total purchase is **greater than $100**, the customer gets a **20% discount**.

2.If the total purchase is between **$50 and $100**, the customer gets a **10% discount**.

3.If the total purchase is less than **$50**, there is **no discount**.

Lets code it

LOOP



Scenario: Daily Step Tracker

Imagine you are developing a step tracker app that encourages users to achieve their daily goal of **10,000 steps**. The app checks the steps entered by the user at regular intervals and provides feedback until the goal is reached

Lets code it

SUB-PROGRAM (Function)

- Functions are reusable blocks of code designed to perform a specific task

How Functions Work

1.Definition: Use the def keyword to define a function.
2.Parameters: Specify inputs in parentheses (optional).
3.Body: Contain the code to execute.
4.Return: Use return to send a result back to the caller (optional)

SUB-PROGRAM (Function)

Built-In vs. User-Defined Functions

Built-In

```
nums = [1, 2, 3, 4]
print(len(nums))
# Output: 4
```

User-Defined

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

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

Problem: Calculate the Balance After Multiple Transactions

Imagine you're building a system to manage a user's bank account balance. The user deposits or withdraws money multiple times throughout the day. Your task is to:

- Calculate the final balance after a series of transactions.
- Ensure the balance never goes below zero (if a withdrawal exceeds the available balance, it should be prevented).

Problem: Calculate the Balance After Multiple Transactions

Algorithm Steps:

Input:

Take the initial balance as input.
Take the number of transactions as input.
For each transaction, take the type of transaction (deposit or withdrawal) and the amount involved.

Process Transactions:

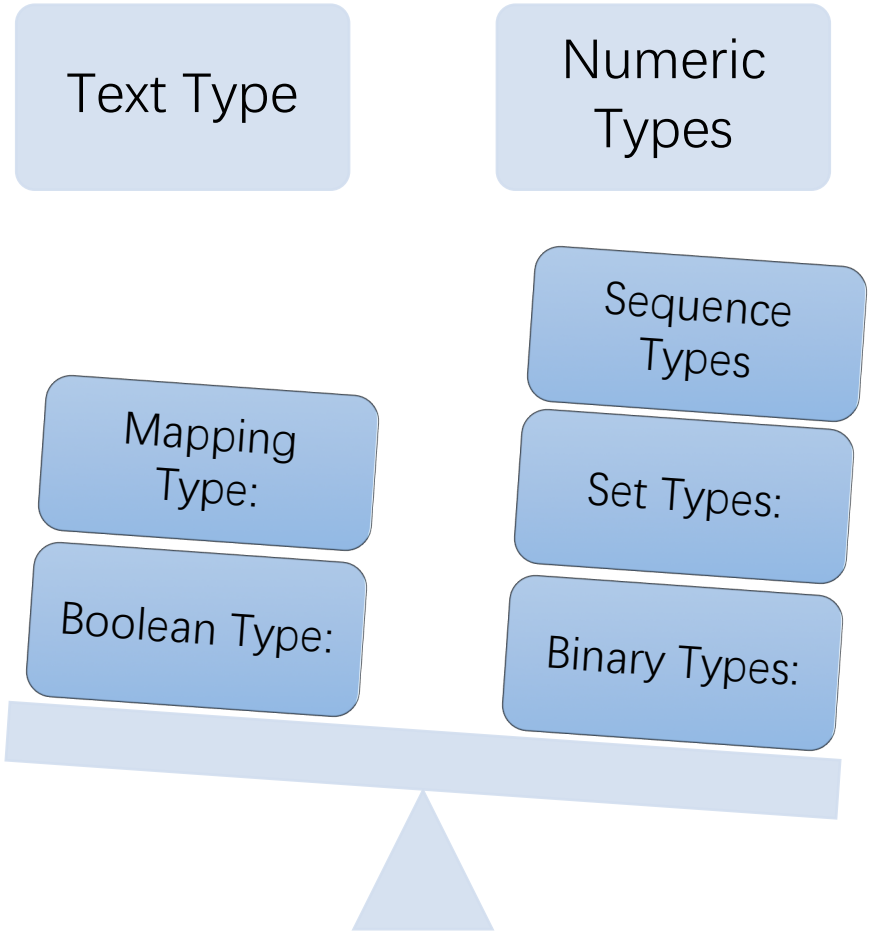
If the transaction is a deposit, increase the balance by the transaction amount.
If the transaction is a withdrawal, check if the balance is sufficient:
If there are enough funds, decrease the balance by the transaction amount.
If not, reject the withdrawal and print a warning.

Output:

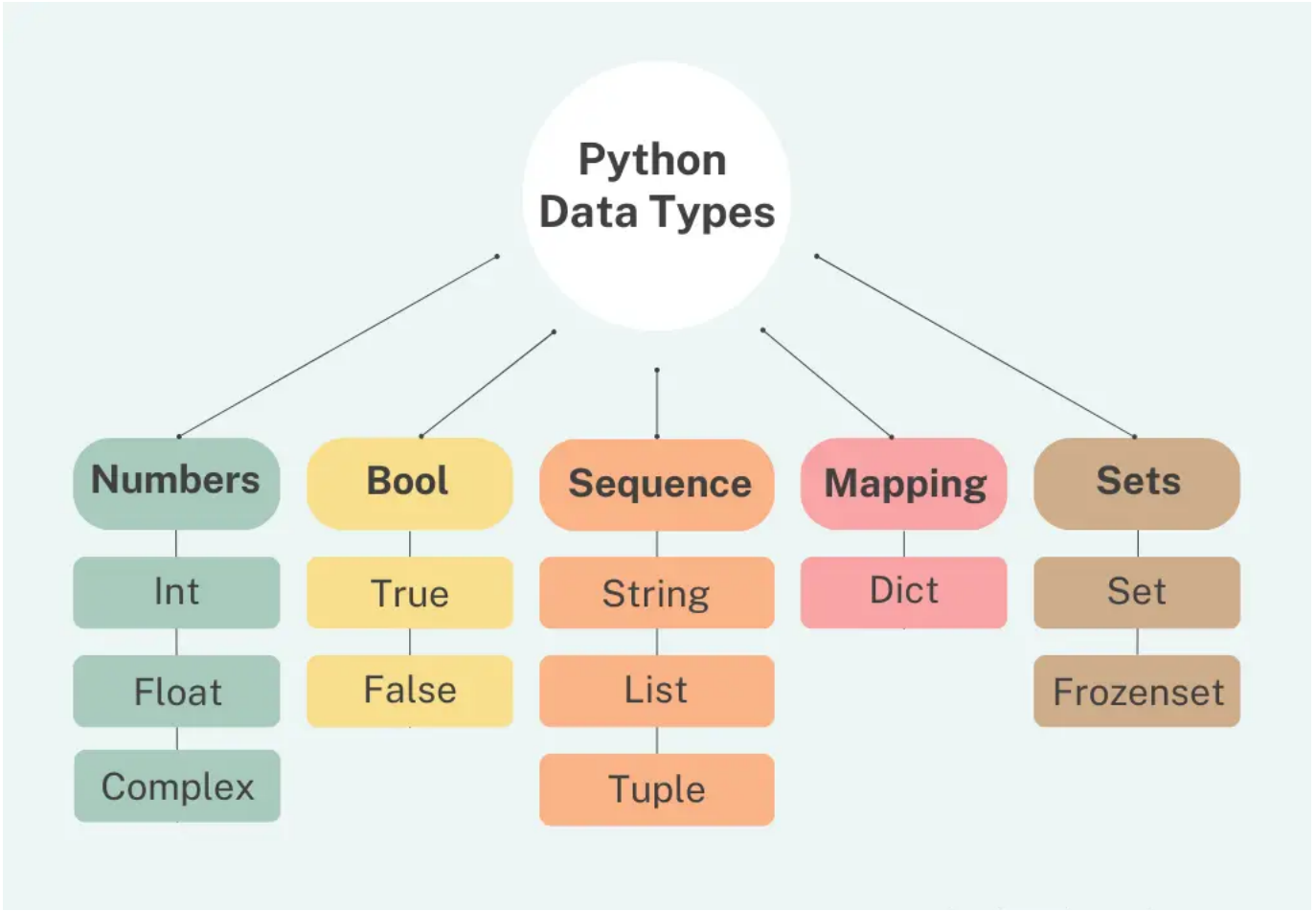
After processing all transactions, print the final balance

Lets code it

Data Types in Python



Data Types in Python



Data Types in Python

Lists

- Lists are one of 4 built-in data types in Python used to store collections of data
- List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1] etc.
- List items can be of any data type
- A list can contain different data types
- **List() constructor**

Tuple

- A tuple is a collection which is ordered and **unchangeable**.

- Tuple items are ordered, unchangeable, and allow duplicate values.

```
fruits = ("apple", "banana", "cherry")
```

Data Types in Python

Set

- Once a set is created, you cannot change its items, but you can remove items and add new items.
- Sets are written with curly brackets

Unordered

- Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable

- Set items are unchangeable, meaning that we cannot change the items after the set has been created.

Duplicates Not Allowed

- Sets cannot have two items with the same value.

Data Types in Python

Dictionary

- To store data values in key:value pairs.

Ordered or Unordered?

- Python version 3.7 = Ordered
- Python version 3.6 and earlier = Unordered

Changeable

- Dictionary are changeable, meaning that we change the items after it has been created.

Duplicates Not Allowed

- Dictionary cannot have two items with the same key.

Questions?

zahmaad.github.io