ENGR 1330: Computational Thinking with Data Science

Lesson 3: Data Structures and Conditional Statements

Dinesh S. Devarajan
Whitacre College of Engineering
Texas Tech University

Topic Outline

- Data Structures in Python

- Conditional Statements in Python

Objectives

- To understand different data structures available in Python that can be helpful in handling data effectively

- To understand and implement the concept of decision making in Python using conditional statements

Computational Thinking Concepts

Data structures:
Arrays, lists, tuples, sets, and dictionaries

⇨ **Data representation**

Data interpretation, manipulation, and analysis of NumPy arrays

⇨ **Decomposition**

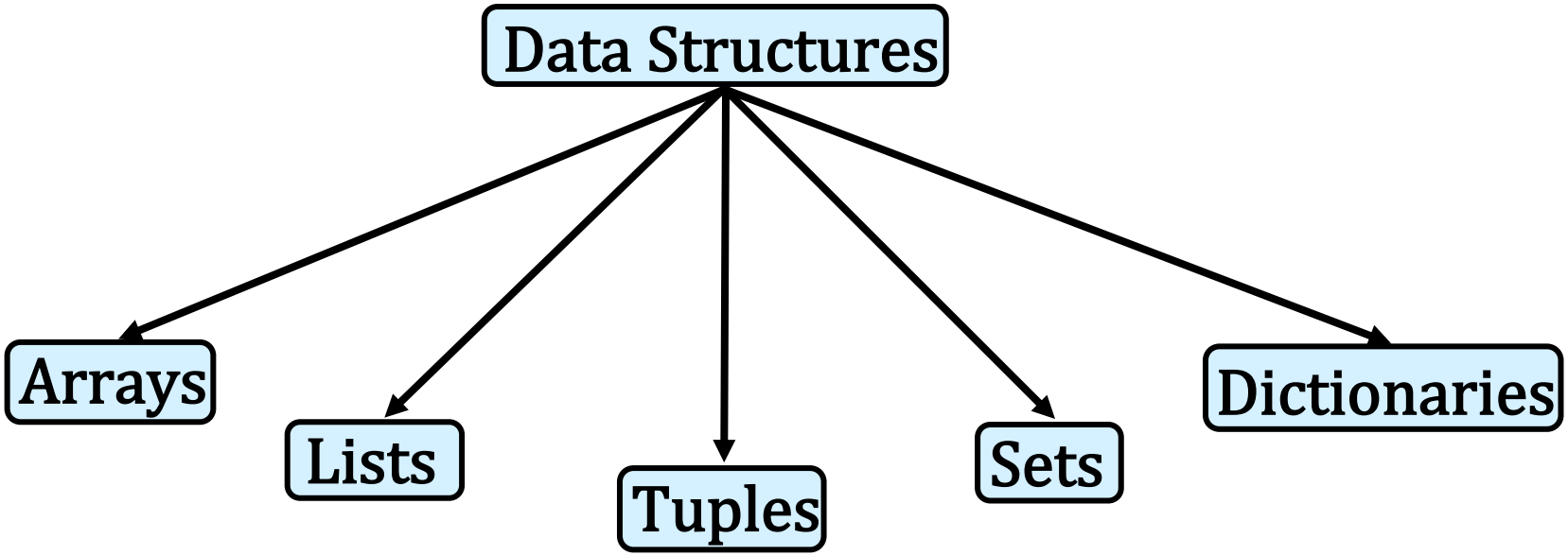
Conditional statements

⇨ **Decomposition**

⇨ **Algorithm design**

Data Structures in Python

Python Data Structures



Arrays

- Arrays: Can be used to store only elements of a specific data type

- Properties:

  - ✓ Ordered: Elements in an array can be indexed

  - ✓ Mutable: Elements in an array can be altered

Arrays

- Visual representation of an array

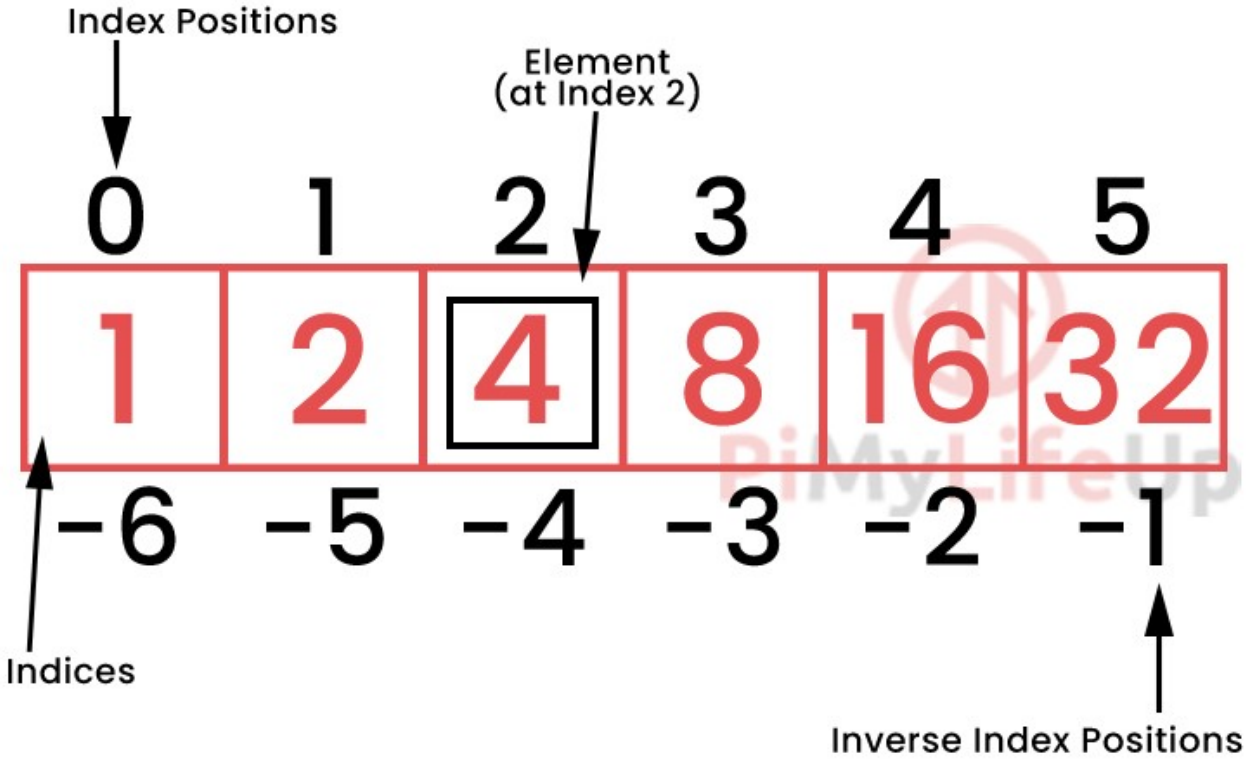


Figure Source: https://pimylifeup.com/python-arrays/

Arrays

- Data type that an array must hold is specified using the type code
  - ✓ ‘f’ for float
  - ✓ ‘d’ for double

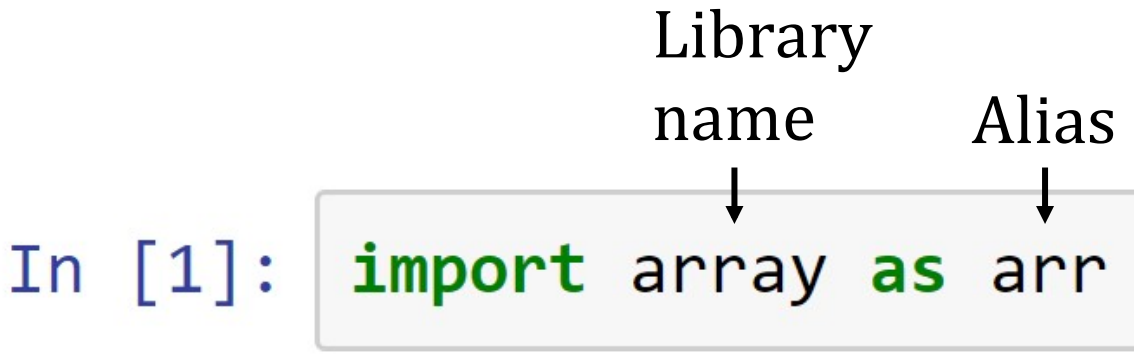
  } Floating numbers

  - ✓ ‘i’ for signed int
  - ✓ ‘I’ for unsigned int

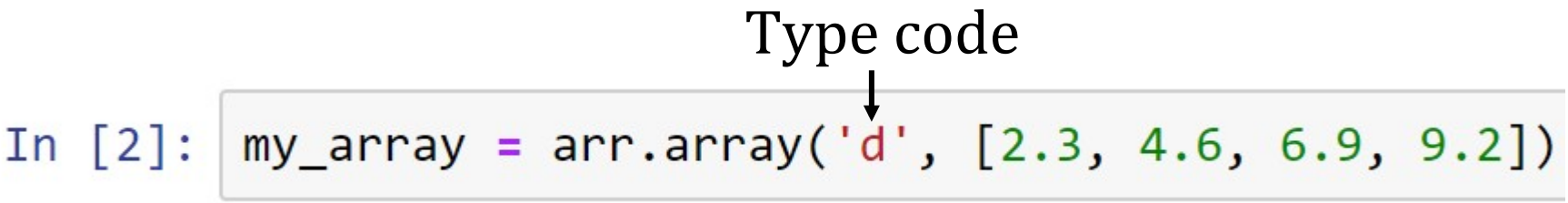
  } Integers

Arrays

- To use arrays, a library named ‘array’ must be imported



- Creating an array that contains floating numbers



(Demo on arrays)

Lists

- Lists: Can be used to store elements of different data types

- Properties:
  - ✓ Ordered: Elements in a list can be indexed
  - ✓ Mutable: Elements in a list can be altered

- Mathematical operations must be applied to each element of the list

Lists

- Elements of a list are enclosed in square brackets [ ]

- Creating lists that contains different data types

```
In [6]: my_list1 = [2.3, 4.6, 6.9, 54, 1001]

In [7]: my_list2 = [2.3, 4.6, 6.9, 'data', 'memory']

In [8]: my_list3 = [2.3, 4.6, 6.9, [1,3,5], ['science', 'engineering']]
```

Nested list: Lists within a list

(Demo on lists)

Tuples

- Tuples: Can be used to store elements of different data types

- Properties:
  - ✓ Ordered: Elements in a tuple can be indexed
  - ✓ Immutable: Elements in a tuple cannot be altered

- Tuples are memory efficient because of their immutable property

Tuples

- Elements of a tuple are enclosed in round brackets ( )

- Creating tuples that contains different data types

```
In [9]: my_tuple1 = (2.3, 4.6, 6.9, 54, 1001)

In [10]: my_tuple2 = (2.3, 4.6, 6.9, 'data', 'memory')

In [11]: my_tuple3 = (2.3, 4.6, 6.9, (1,3,5), ('science', 'engineering'))
```

Nested tuple: Tuples within a tuple

(Demo on tuples)

Sets

- Sets: Can be used to store elements of different data types

- Properties:
  - ✓ Unordered: Elements in a set cannot be indexed
  - ✓ Mutable: Elements in a set can be altered
  - ✓ Non-repetition: Elements in a set are unique

Sets

- Elements of a set are enclosed in curly brackets { }

- Creating sets that contains different data types

```
In [12]: my_set1 = {2.3, 4.6, 6.9, 54, 1001}

In [13]: my_set2 = {2.3, 4.6, 6.9, 'data', 'memory'}
```

- Sets cannot be nested

(Demo on sets)

Dictionaries

- Dictionaries: Can be used to store elements of different data types

- Each element in a dictionary have unique key

- Properties:
  - ✓ Unordered: Elements in a dictionary cannot be indexed
  - ✓ Mutable elements: Elements in a dictionary can be altered
  - ✓ Immutable keys: Keys in a dictionary cannot be altered

Dictionaries

- Key-Element pairs of a dictionary are enclosed in curly brackets { }

- Creating dictionaries that contains different data types

```
In [15]: my_dict1 = {'Name': 'Joe', 'Age': 8, 'Class': 'Second'}

In [16]: my_dict2 = {'Brand': 'Ford', 'Model': 'Mustang', 'Year': 2020}

In [17]: my_dict3 = {'key': {'innerkey': [1, 2, 3]}}
```

Nested dictionary: Dictionary within a dictionary

(Demo on dictionaries)

Conditional Statements in Python

Conditional Statements

- Decision making via conditional statements is an important step in algorithm design

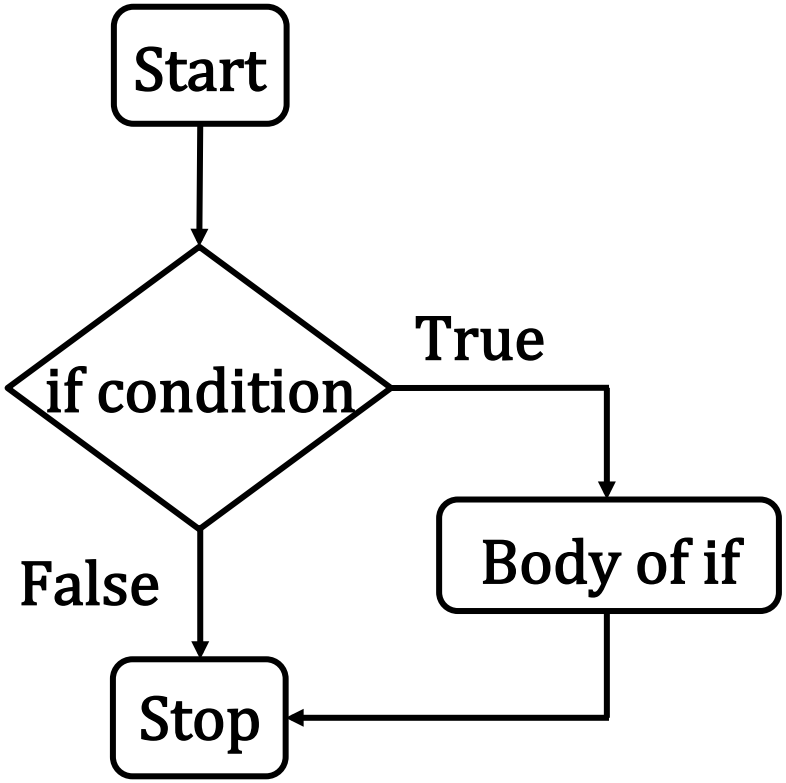
- Controls the flow of execution of a program

Conditional Statements

- Conditional statements in Python
  - ✓ if statement
  - ✓ if....else statements
  - ✓ if....elif....else statements

if Statement

- if: To check if a condition is true or not



if Statement

- Implementation in Python

```
if <condition>:
    <statements
    >
```

- Indentation is required for statements inside the body of ‘if’ statement

- Statements intended by four spaces

Discussion Exercise

- What would be the outputs for the below two 'if' statements?

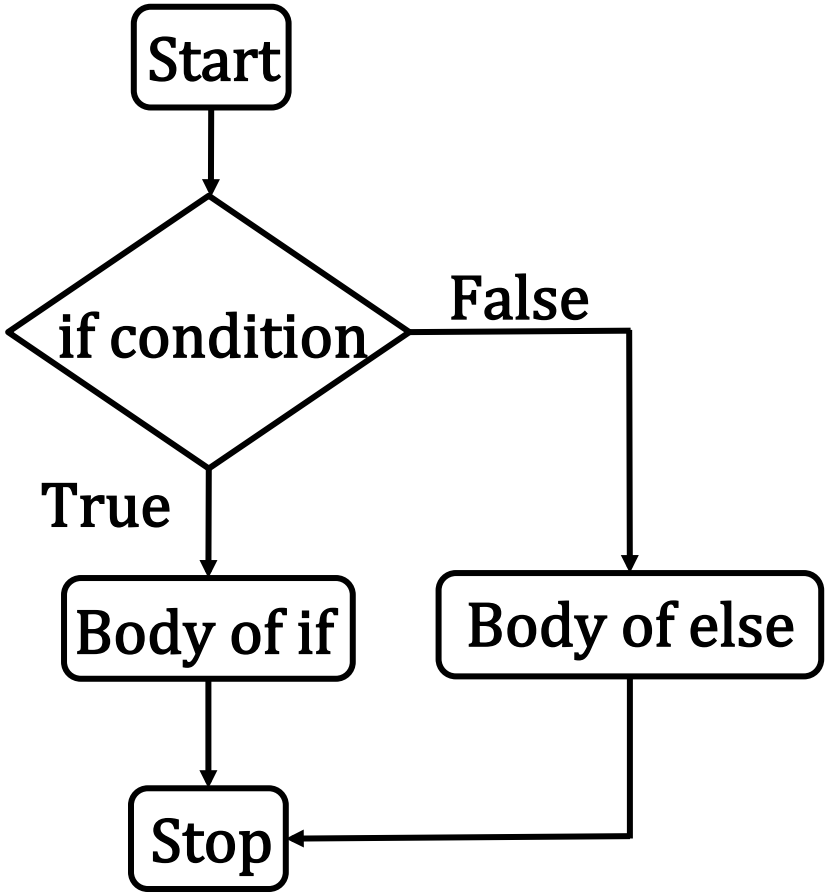
```
In [21]: if 100>200:
             print('Larger')
```

```
In [279]: if 100<200:
              print('Smaller')
```

(Demo)

if....else Statements

- if....else: Program proceeds to 'else' block only if the condition associated with 'if' block fails



if....else Statements

- Implementation in Python

```
if <condition>:
    <statements>
else:
    <statements>
```

- Note: if and else are two separate blocks of code

Discussion Exercise

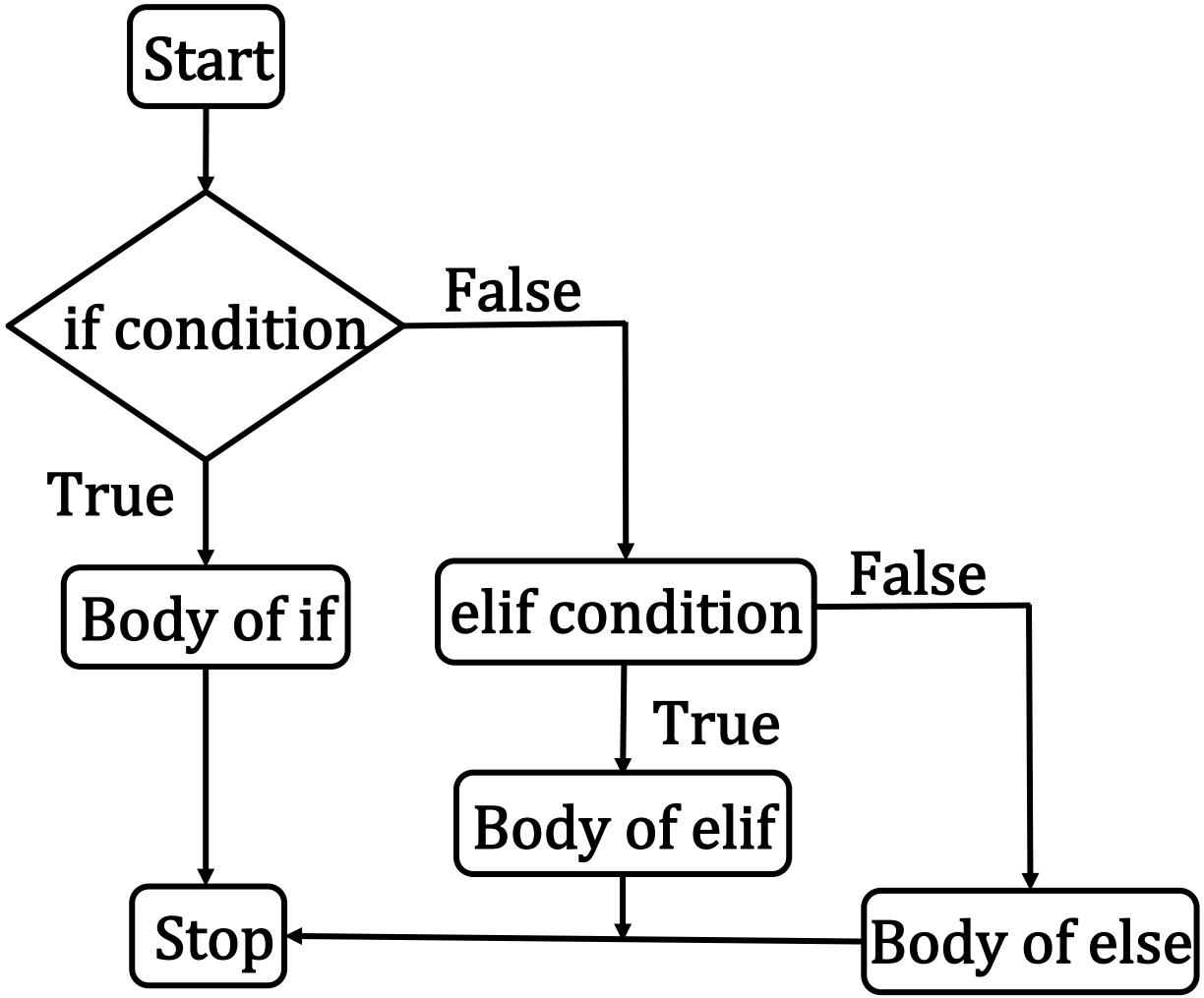
- What would be the output for the below 'if....else' statements?

```
In [11]: if ('bob' in ['sam', 'bob', 'mary', 'joe']):
             print('Present')
         else:
             print('Absent')
```

(Demo)

if....elif....else Statements

- if....elif....else: Used for evaluating multiple conditions



if....elif....else Statements

- Implementation in Python

```
if <condition>:
        <statements>
elif <condition>:
        <statements>
else:
        <statements>
```

- Note: if, elif, and else are three separate blocks of code

Discussion Exercise

- What would be the output for the below 'if....elif....else' statements?

```
In [22]: num = 3.4

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

(Demo)

Nested Conditional Statements

- What would be the output for the below nested conditional statements?

```
In [102]: marks = 75

if marks >= 55:
    if marks >= 55 and marks <= 69:
        print("D grade")
    elif marks >=70 and marks <= 79:
        print("C grade")
    elif marks >=80 and marks <= 89:
        print("B grade")
    else:
        print("A grade")
else:
    print ("Fail")
```

if....elif....else statements within an if statement

(Demo)

Summary

- Concepts of different data structures in Python are covered

- Concepts of conditional statements in Python are covered