PLAN 396
Lecture 12

Dr. Hossen Asiful Mustafa

https://hossenmustafa.buet.ac.bd

Introduction

- Sequences
  - Also know as arrays in other languages
  - Store related data types
  - 3 types
    - Strings
    - Lists
    - Tuples
- Mappings
  - In most languages called hashes or associative arrays
  - Dictionaries are the only python mapping container

Creating Sequences

- Strings
  - Use quotes
  - **`string1 = "‶`**
- Tuples
  - Use parenthesis
  - Separate multiple items with a comma
  - **`tuple = ()`**
  - **`singleton = 1,`** - this is a one element tuple or a singleton
- Lists
  - Use brackets
  - Separate multiple items with a comma
  - **`list1 = []`**

Using Lists

- Creating a List

```
anylist = []
chevycars = [“Corvette”, “Aveo”, “Malibu”, “Cobalt”,
  “Impala”, “Silverado”, “Trailblazer”, “Tahoe”,
  “Colorado”, “Suburban”]
```

- Length Function

```
num_cars = len( chevycars )
```

- in Operator

```
if “Corvette” in chevycars:
  print “That’s a nice car!!!!”
```

Using Lists

- Indexing

```
print chevycars[4]
```

- Slicing

```
print chevycars[0:4]
```

- Concatenation

```
chevycars += [ “SSR”, “Monte Carlo”,
  “Avalanche”, “Equinox”, “Blazer”,
  “Astro”, “Uplander”, “Venture” ]
```

Using Lists

- Lists are mutable
  - The elements in a list can change value
- Change by assignment

```
xmaslist[0] = “Black Corvette with optional blonde”
xmaslist[1] = “Trip to Las Vegas”
```

- Change by slicing

```
inventory[4:6] = “Orb of Future Telling”
```

Using Lists

- Deleting a list element

```
del inventory[2]
```

- Deleting a list slice

```
del inventory[4:6]
```

List Methods

- append(value)
  - Adds value to the end of the list
- sort()
  - Sorts the elements, smallest value first
- reverse()
  - Reverses the order of a list
- count(value)
  - Returns the number of occurrences of value in the list

List Methods

- index(value)
  - Returns the first position number of where value occurs
- insert(i, value)
  - inserts value at position I
- pop([i])
  - Returns value at position i and removes value from the list. Providing the position number is optional. Without it, the last element in the list is removed.
- remove(value)
  - Removes the first occurrence of value from the list.

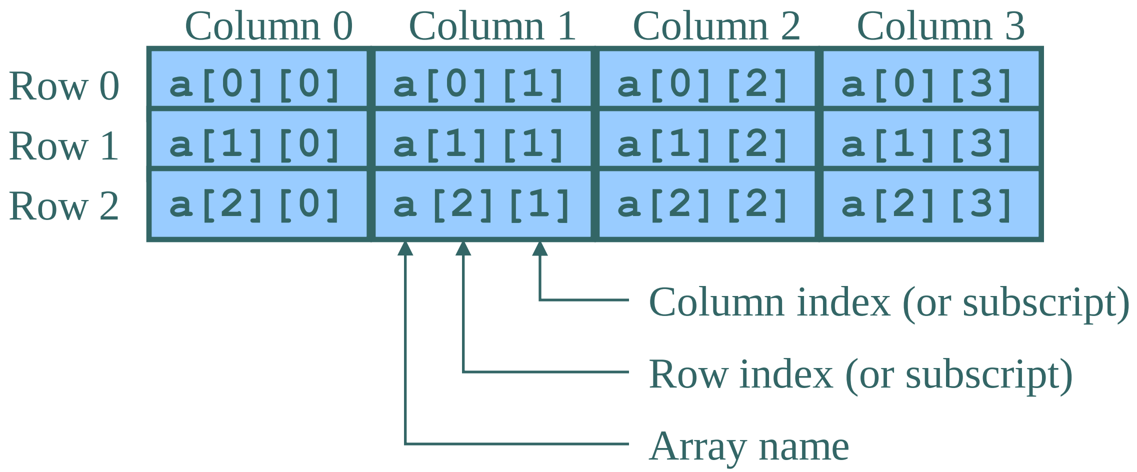
Tuples vs. Lists

- Tuples are faster than lists. Because the computer knows they will not change, tuples can be stored in a way that makes using them faster that using lists. For simple programs, this speed difference will not matter, but in more complex applications, with very large sequences of information, it could.
- Tuples' immutability makes them perfect for creating constants since they cannot change. Using tuples can add a level of safety and clarity to your code.
- Sometimes tuples are required. In some cases Python requires immutable values.

Nested Sequences

- Tuples and Lists can be sequences of anything
  - Including other tuples and lists
- Nested Sequences
  - Sequences inside other sequences
  - A great method to organize complex collections of information
- Common use is a table
  - By convention it is organized by row and column

Nested Sequences



	Column 0	Column 1	Column 2	Column 3
Row 0	`a[0][0]`	`a[0][1]`	`a[0][2]`	`a[0][3]`
Row 1	`a[1][0]`	`a[1][1]`	`a[1][2]`	`a[1][3]`
Row 2	`a[2][0]`	`a[2][1]`	`a[2][2]`	`a[2][3]`

Nested Sequences

- Creating nested sequences (examples)

```
nested_1 = [“first”, (“second”, “third”), [“fourth”, “fifth”,
   “sixth” ]]
scores = [(“Moe”, 1000),(“Larry”, 1500),(“Curly”, 2000)]

nested_2 = (“deep”,(“deeper”,(“deepest”,”still deepest”)))
```

Nested Sequences

- Accessing Nested Sequences (example)

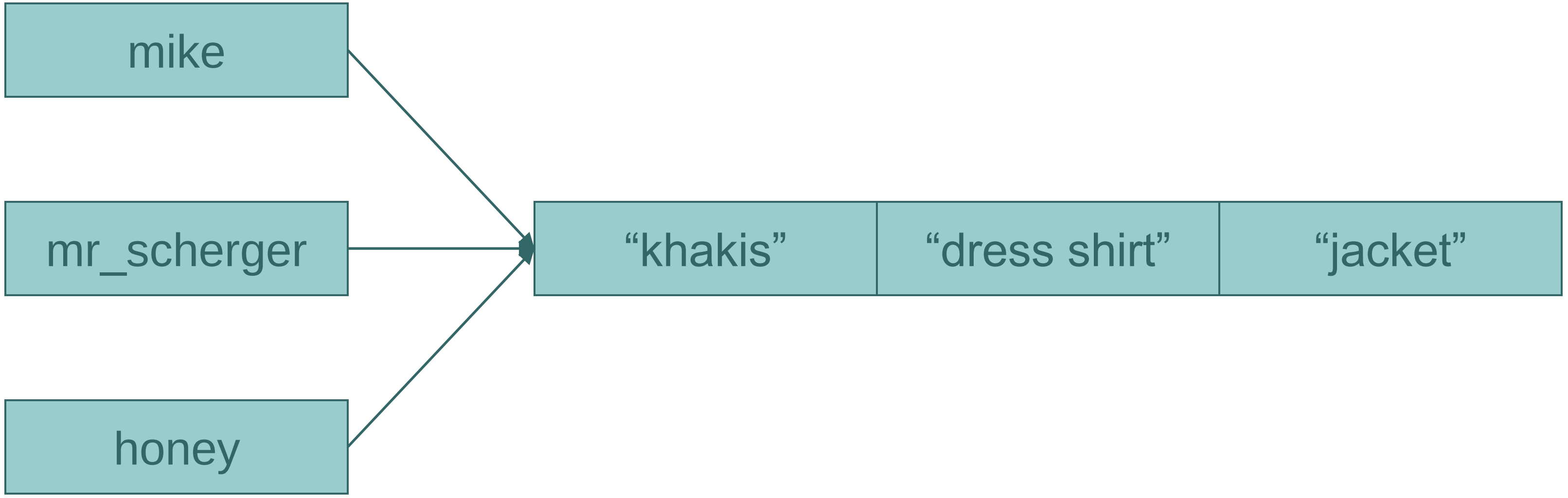
```
>>> scores = [(“Moe”, 1000),(“Larry”, 1500),(“Curly”, 2000)]
>>> print scores[0]
(‘Moe’, 1000)
>>> print scores[1]
(‘Larry’, 1500)
>>> print scores[2]
(‘Curly’, 2000)
>>> a_score = scores[2]

>>> print scores[2][0]
Curly
>>> print scores[2][0][4]
y
```

Shared References

```
>>> mike = ['khakis', 'dress shirt', 'jacket']
>>> mr_scherger = mike
>>> honey = mike
>>> print mike
['khakis', 'dress shirt', 'jacket']
>>> print mr_scherger
['khakis', 'dress shirt', 'jacket']
>>> print honey
['khakis', 'dress shirt', 'jacket']
>>>
```

Shared References



Class Assignment

- Write a program named classassignment15.py
- The program should:
  - Create a list with 5 integer elements getting input from user
  - Generate a new list with item between 1-10 where the item isn’t in the 1st List
  - Example
    - L1 = [1,2,3, 9,5]
    - L2 = [4,6,7,8,10]
    - Print all the lists
    - Sort both lists and print in ascending and descending order
    - Merge the lists and print in ascending and descending order