# Lists, Tuples and Dictionaries

HORT 59000

Lecture 10

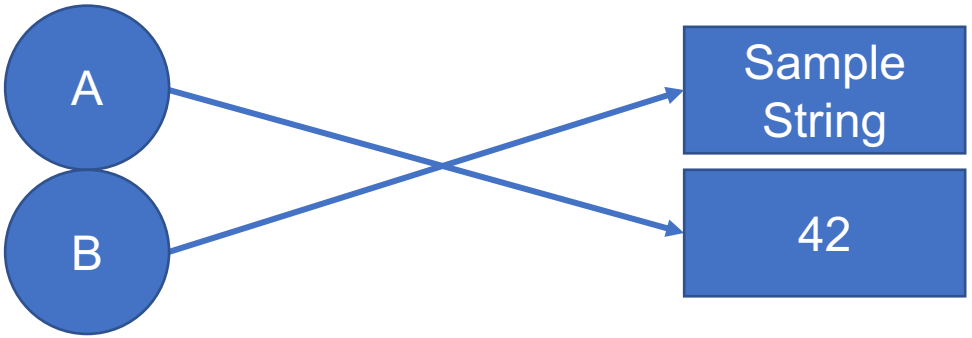
Instructor: Kranthi Varala

# Core data types

- Numbers
- Strings
- Lists
- Dictionaries
- Tuples
- Files
- Sets

# References and dynamic typing

- Dynamic typing allows changing the type of a variable.
- `A = '42'` now changes the apparent data type of A to an integer.




- The reference from A to 'Sample String' is removed.
- B still points to the 'Sample String' object.
- If all variable reference are removed from an object, the object will be marked for removal by Python.
- The process of removing dereferenced objects is called garbage collection

# Lists

- List is a general sequence object that allows the individual items to be of different types.
- Equivalent to arrays in other languages.
- Lists are mutable, i.e., a list can be changed without having to create a new list object

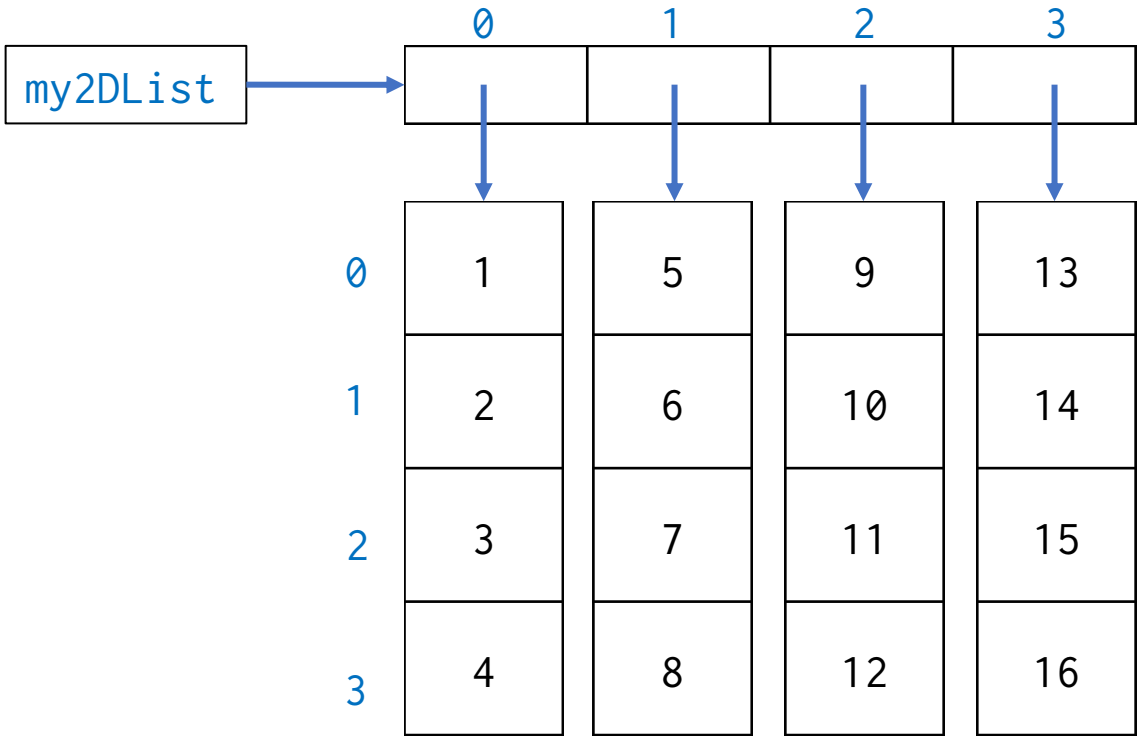
```
>>> L = [25,1,2,15]
>>> L
[25, 1, 2, 15]
```

# Lists: Common Methods

- L.append() : Adds one item to the end of the list.
- L.extend() : Adds multiple items to the end of the list.
- L.pop(i) : Remove item ‘i’ from the list. Default:Last.
- L.reverse() : Reverse the order of items in list.
- L.insert(i,item): Inserts ‘item’ at position i.
- L.remove(item) : Finds ‘item’ in list and deletes it from the list.
- L.sort(): Sorts the list in- place i.e., changes the sequence in the list.

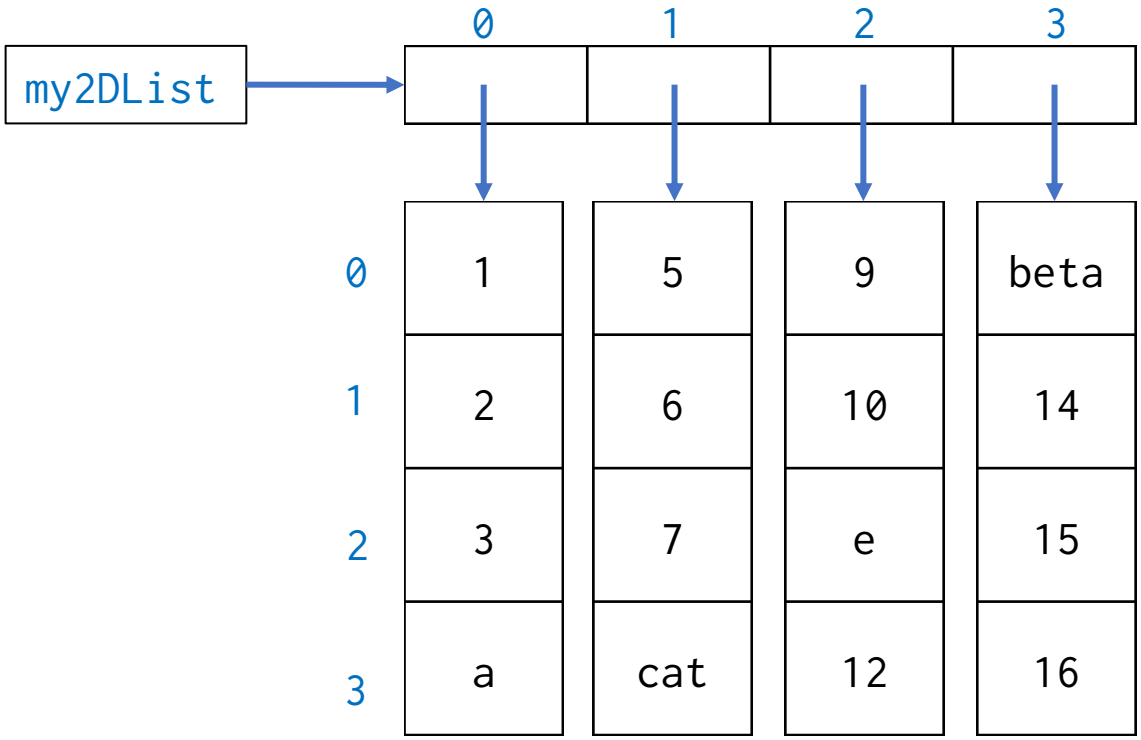
# MultiDimensional Lists

- Lists are of arbitrary length and and easily be nested.
- Simplest nested lists are 2 –dimensional matrices.
- `my2DList = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]`



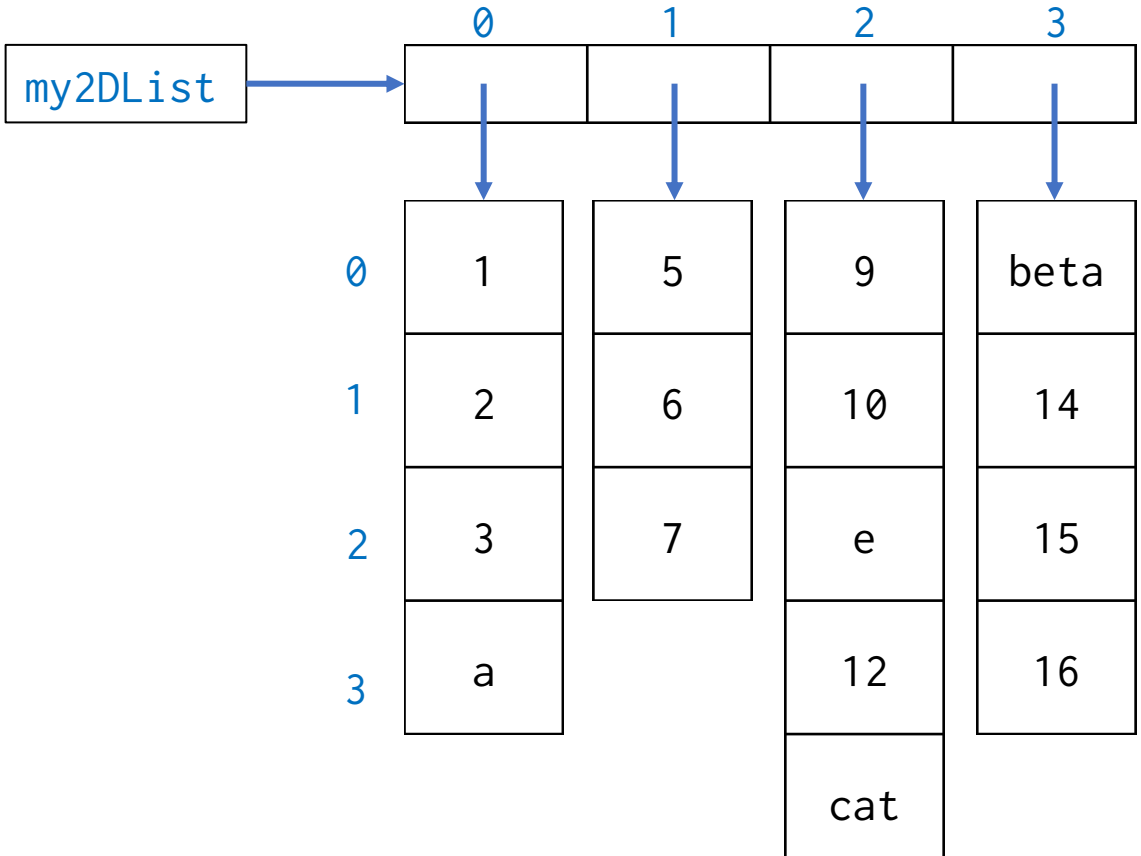
# MultiDimensional Lists

- Nested Lists need not be homogeneous.
- `my2DList = [[1,2,3,'a'],[5,6,7,'cat'],[9,10,'e',12],['beta',14,15,16]]`



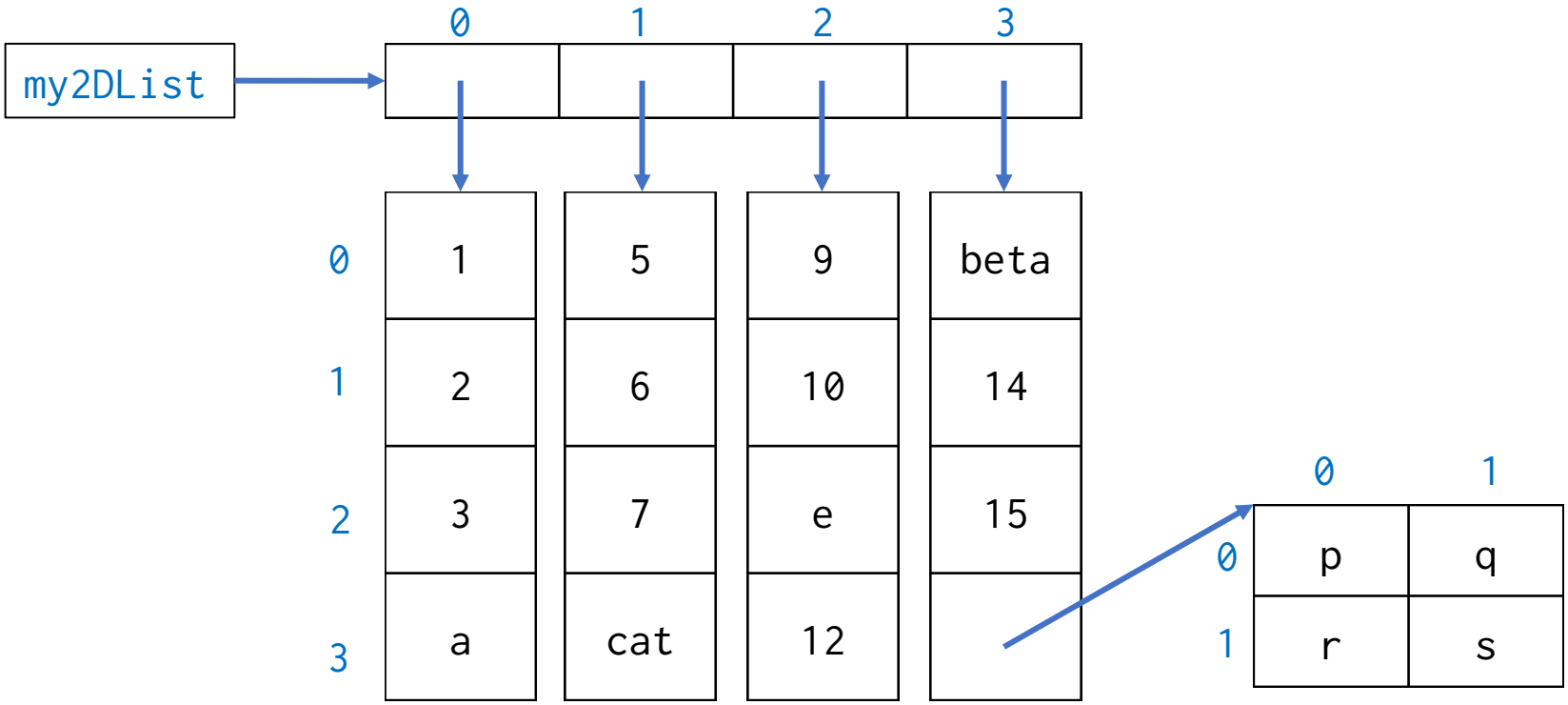
# Arbitrary dimensional Lists

- Nested Lists need not be of the same length.
- `my2DList = [[1,2,3,'a'],[5,6,7],[9,10,'e',12,'cat'],['beta',14,15,16]]`



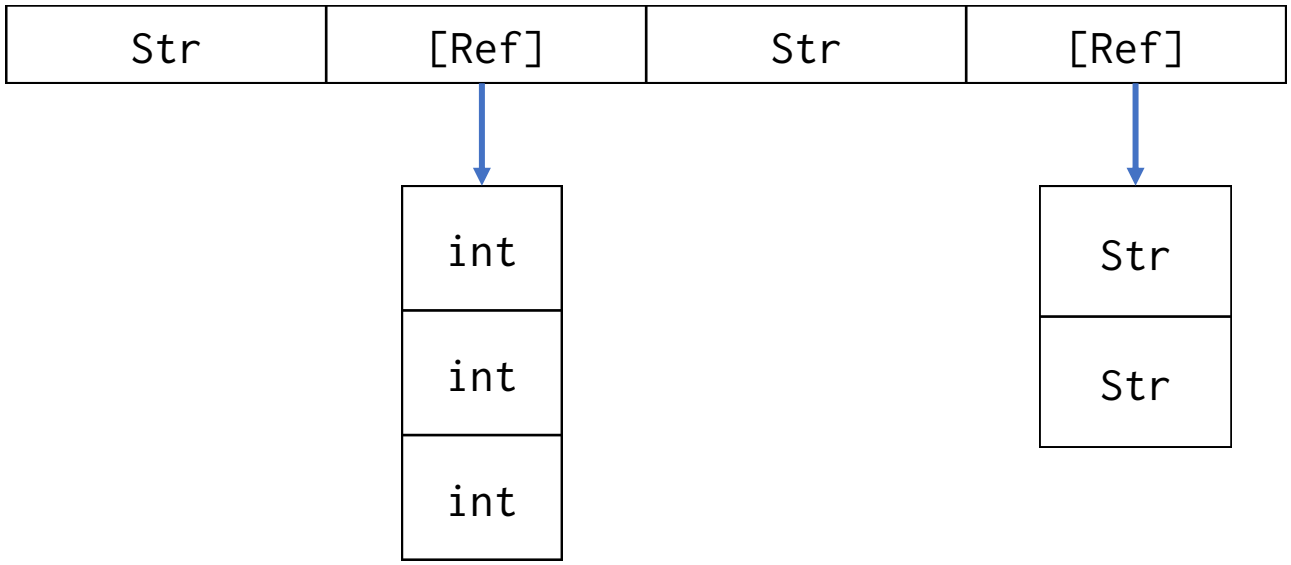
# Arbitrary dimensional Lists

- Nested Lists can have arbitrary depth as well.
- `subL = [['p','q'],['r','s']]`
- `my2DList = [[1,2,3,'a'],[5,6,7,'cat'],[9,10,'e',12],['beta',14,15,subL]]`



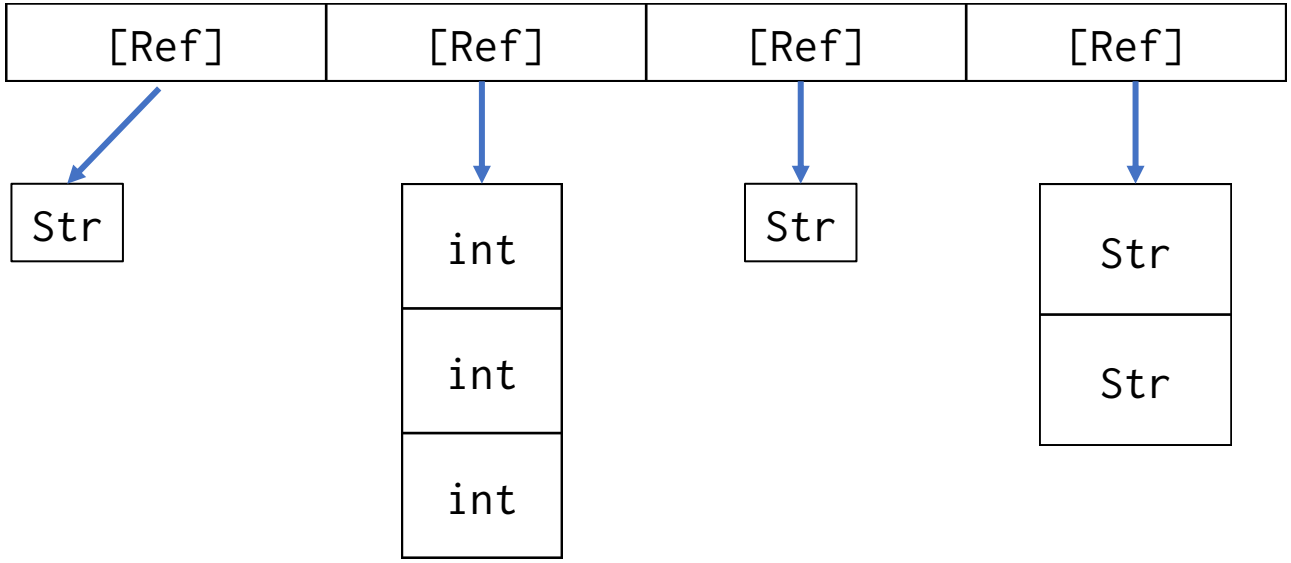
# Lists as sequences of references

- `myList = ['Name',[Month,Date,Year],Address,[Home,Cell]]`



# Lists as sequences of references

- `myList = ['Name',[Month,Date,Year],Address,[Home,Cell]]`



# Lists are mutable!!

```
>>>subL = [[’p’,’q’],[‘r’,’s’]]

>>>myList = [[1,2,3,’a’],[5,6,7,’cat’],[9,10,’e’,12],[’beta’,14,15,subL]]

>>>myList

[[1,2,3,’a’],[5,6,7,’cat’],[9,10,’e’,12],[’beta’,14,15,[[‘p’,’q’],[‘r’,’s’]
]]]

>>>subL[0][1] = ‘z’

>>>myList


[[1,2,3,’a’],[5,6,7,’cat’],[9,10,’e’,12],[’beta’,14,15,[[‘p’,’z’],[‘r’,’s’]
]]]
```

# Tuples

- Tuples are immutable general sequence objects that allows the individual items to be of different types.
- Equivalent to lists, except that they can't be changed.

```
>>> myTuple=('0','a','bob',4.89)
>>> myTuple
('0', 'a', 'bob', 4.889999999999997)
>>> myTuple[1]
'a'
>>> myTuple[1:3]
('a', 'bob')
```

# Tuples

- Tuple.count(value) : Returns number of occurences of value.
- Tuple.index(value,[start,stop]) : Returns first index of value.
- Typically used to maintain data integrity within the program.

```
>>> myTuple= (0,)
>>> myTuple
(0,)
>>> myTuple= 0,
>>> myTuple
(0,)
```

Even single elements need a comma

Parentheses () are optional

# Dictionaries

- Dictionaries are unordered collections of objects, optimized for quick searching.
- Instead of an index, objects are identified by their ‘key’.
- Each item within a dictionary is a ‘key’:’value’ pair.
- Equivalent to hashes or associative arrays in other languages.
- Like lists and tuples, they can be variable-length, heterogeneous and of arbitrary depth.
- ’Keys’ are mapped to memory locations by a hash function

# Hash function

- A hash function converts a given key value into a ‘slot’ where its value will be stored.
- A hash function always takes a fixed amount of time and always returns the same slot for the same ‘key’.
- When program searches for a ‘key’ in the dictionary, the slot it should be in is calculated and the value in it, if any, is returned.
- Creating hashes is expensive, searching is cheap.

# Dictionaries

```
>>> myDict={}
```

```
>>> myDict={'name':'Bob','surname':'Smith','age':40}
>>> myDict
{'age': 40, 'surname': 'Smith', 'name': 'Bob'}
>>> myDict['name']
'Bob'
```

```
>>> myDict['YOB'] = 1970
>>> myDict
{'age': 40, 'surname': 'Smith', 'name': 'Bob', 'YOB': 1970}
>>> myDict['name'] = 'Robert'
>>> myDict
{'age': 40, 'surname': 'Smith', 'name': 'Robert', 'YOB': 1970}
```

# Dictionaries: Common Methods

- D.keys() : List of keys
- D.values() : List of values
- D.clear() : remove all items
- D.update(D2) : Merge key values from D2 into D. NOTE: Overwrites any matching keys in D.
- D.pop(key) : returns the value of given key and removes this key:value pair from dictionary.

# Summary: Lists vs. Tuples vs. Dictionaries

- All three data types stores a collection of items.
- All three allow nesting, heterogeneity and arbitrary depth.
- Choice of data type depends on intended use:
  - Lists : Best suited for ordered collections of items where the order or the items themselves may need to be changed.
  - Tuples: Best suited for maintaining a copy of the collection that will not be accidentally changed during the program.
  - Dictionary : Best suited for storing labeled items, especially in collections where frequent searching is required.