Lists

Chapter 04a

Python Collection Data Types

- LIST
- TUPLE
- SET
- DICTIONARY

Python Collection Data Type: List

- A LIST contains multiple values in a ordered sequence and can be changed.
- Enclosed in [ ].
- Used to store a collection of data.
- **TIP:** Keep values in a list of the same data type.
- However, values in lists can be different types, NOT RECOMMENDED.

Examples for a LIST of values

- Scores in all courses taken
- Shopping List
- Athletic team roster
- Guest list for a wedding
- Names in a phone book

List Syntax

empty_list = [ ]

list_name = [item1, item2, … separated by commas]

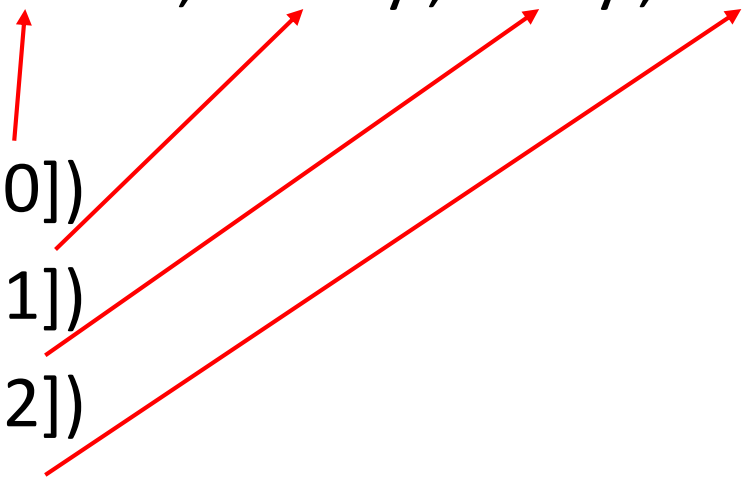
guests = [‘Michael’, ‘Sandy’, ‘Kelly’, ‘Joe’]

You can create an empty list and add values to it.

```
guest = []
scores = []
animals = []
```

List Syntax

guests = [‘Michael’, ‘Sandy’, ‘Kelly’, ‘Joe’]



print(guests[0])

print(guests[1])

print(guests[2])

print(guests[3])

#values in list begins with index 0.

#index must be integers

#Error if index number exceeds list

LIST Methods

For Lists

- listName.**index**(listItem) – **SEARCH** the list and return the position where it was found.
- listName.**append**(listItem) – **ADDING** an item to the end of the list
- listName.**insert**(index#,listItem) – **INSERTING** an item to a specific position in the list
- listName.**remove**(listItem) – **REMOVING** item from list
- listName.**pop**(listIndex#) – **REMOVING** item from list
- **del** listName[indexNum] – **DELETING** item from list

Accessing an Item in a List by specifying it's position in the list (called the INDEX)

```
animals = ['cat', 'bat', 'rat']
```

```
print(animals[0])
```
OUTPUT: cat

```
print(animals[1])
```
OUTPUT: bat

Accessing an Item in a List by their INDEX

EXAMPLE:

```
scores = [78, 85, 62, 49, 98]          #scores is the list

print(scores)                          #[78, 85, 62, 49, 98]
print(scores[2])                       #62
print(scores[1] + scores[2])           #147
```

You can even access a list backwards (Negative Indexes)

```
animals = [‘cat’, ‘bat’, ‘rat’]

print(animals[-1])

print(animals[-2])
```

#**REMINDER:** values in list begins with index 0.

OUTPUT: rat

#-1 refers to last index in list

OUTPUT: bat

#-2 refers to 2nd to last index in list

List ERRORS

```
animals = [‘cat’, ‘bat’, ‘rat’]
```

#**REMINDER:** values in list begins with index 0.

```
print(animals[1.0])
```

#ERROR: index must be INTEGER

```
print(animals[9])
```

#ERROR: index cannot exceed range of list

Finding a Value in a List - index() Method

- If value is in the list, **index() method** returns position (index number) if where the value was found in the list.
- If value is not in the list, returns Error
- If duplicate values, the index of the first appearance is returned.

animals = [‘bat’, ‘cat’, ‘rat’]

animals.index(‘bat’) #returns 0

animals.index(‘dog’) #returns ValueError

Updating Values in a List

```
animals = [‘cat’, ‘bat’, ‘rat’, ‘bird’]

animals[1] = ‘ant’
print(animals)
```

#you can also use an index of a list to **update the value** at that index

OUTPUT:

[‘cat’, ‘ant’, ‘rat’, ‘bird’]

Adding Values to the end of a List - append() Method

append method adds to **end** of a list.

```
animals = ['bat', 'cat', 'rat']
print(animals)
```

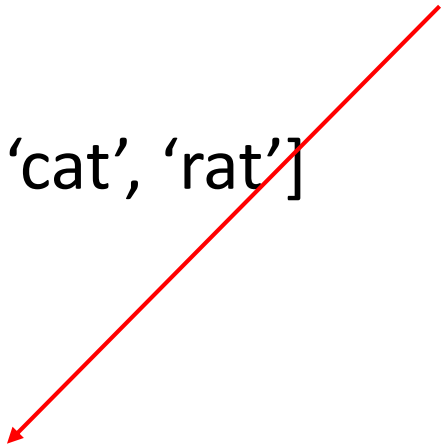
OUTPUT: ['bat', 'cat', 'rat']

```
animals.append('dog')
print(animals)
```

OUTPUT: ['bat', 'cat', 'rat', 'dog']

Inserting Values to a specific index in a List - insert() Method

insert method adds to **a specific index** in a list.



```
animals = [‘bat’, ‘cat’, ‘rat’]
print(animals)
```

OUTPUT: [‘bat’, ‘cat’, ‘rat’]

```
animals.insert(1, ‘dog’)
print(animals)
```

#Adds to the 2nd item in the list

OUTPUT: [‘bat’, ‘dog’, ‘cat’, ‘rat’]

REMOVE a Value from a List - remove() Method

- If value is in the list, remove() method **removes value** from list.
- If value is not in the list, returns **ValueError**
- If duplicate values, the index of the first appearance is removed.
- Does not leave a blank item in the list if removed or deleted.

REMOVE a Value from a List - remove() Method

Use when you know the value to be removed.

```
animals = ['bat', 'cat', 'rat']
animals.remove('bat')                    #removes the specified item


print(animals)                           #['cat', 'rat']

animals.remove('dog')                    #returns ValueError
print(animals)
```

REMOVE a specified index - pop() Method

Use when you know position in the list.

```
animals = ['bat', 'cat', 'rat']
animals.pop(1)                    #removes item at the specified index


print(animals)                    #['bat', 'rat']
```

REMOVE no index specified- pop() Method

```
animals = [‘bat’, ‘cat’, ‘rat’]
animals.pop()                    #removes last item

print(animals)                   #[‘bat’, ‘rat’]
```

Removing Values from a List – del Statement

*Use when you know the index to be removed.

animals = [‘cat’, ‘bat’, ‘rat’, ‘bird’] [‘cat’, ‘bat’, ‘rat’, ‘bird’]

del animals[2] [‘cat’, ‘bat’, ‘bird’]

Sorts a Value from a List of Numbers or List of Strings - sort() Method

- Sorts uppercase before lowercase.
- If list is combination of numeric and string, returns TypeError

animals = [‘cat’, ‘rat’, ‘bat’]

animals.sort() #[‘bat’, ‘cat’, ‘rat’]

METHOD 1: Iterate over a List

```
animals = [‘cat’, ‘rat’, ‘bat’, ‘bird’]

for pet in animals:
        print(pet)
```

pet
cat
rat
bat
bird

METHOD 2: Iterate over a List using Range

```
animals = [‘cat’, ‘rat’, ‘bat’, ‘bird’]

for pet in range(4):
        print(animals[pet])
```

pet		
0	animals.[0]	cat
1	animals.[1]	rat
2	animals.[2]	bat
3	animals.[3]	bird

METHOD 3: Iterate over a List using len Function

```
animals = [‘cat’, ‘rat’, ‘bat’, ‘bird’]

for pet in range(len(animals)):
        print(animals[pet])
```

pet		
0	animals.[0]	cat
1	animals.[1]	rat
2	animals.[2]	bat
3	animals.[3]	bird

Example: Iterate over a List

```
supplies = [‘pens’, ‘staplers’, ‘binders’, ‘pens’]

for item in range(len(supplies)):
        print(supplies[item])
```

List Concatenation and List Replication

```
numbers = [1, 2, 3]
new_list = numbers + [4,5,6]
print(new_list)                    [1,2,3,4,5,6]

new_numbers = numbers * 3          [1,2,3,1,2,3,1,2,3]
print(new_numbers)
```