Python Lists

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

  scores = [76, 88, 95]

- Shopping List

  grocery = [“bread”, “carrots”, “cheese”]

- Athletic team roster

  roster = [“John”, “Lee”, “Jackie”]

- Guest list for a wedding

  guests = [“Mike”, “Sandy”, “Kelly”]

- Names in a phone book

  names = [“Jane”, “Chris”, “Al”, “Phil”]

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.

```
guests = [ ]
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

guests	guests[0]	guests[1]	guests[2]	guests[3]
	Michael	Sandy	Kelly	Joe

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

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

```
print(animals)
print(animals[0])
print(animals[1])
```

What is the output?

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’]

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

print(animals[-1])

# OUTPUT: rat

#**-1 refers to last index in list**

print(animals[-2])

# OUTPUT: bat

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

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
#using an assignment = statement
```

OUTPUT:

```
[‘cat’, ‘ant’, ‘rat’, ‘bird’]
```

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

METHOD 1: Iterate over a List

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

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

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

METHOD 1: Iterate over a List

supplies	supplies[0]	supplies[1]	supplies[2]	supplies[3]
	pens	staplers	binders	pencils

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

for item in supplies:
        print(item)
```

METHOD 1: Iterate over a List

scores	scores[0]	scores[1]	scores[2]	scores[3]
	80	90	95	75

```
scores = [80, 90, 95, 75]

for num in scores:
        print(num)
```

METHOD 2: Iterate over a List using Range

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

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

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

METHOD 3: Iterate over a List using len Function

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

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

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