GOVERNMENT OF WESTERN AUSTRALIA

TAFE WA South Metropolitan

Data Structures – Lists and Sets

(Session 8)

Review – for Loop

- **for** Loop – execute a block of code (fixed number of times)
- Used for iterating over a sequence. For example:

```
# for is a keyword, in operator is for membership testing
# A colon (:) is mandatory, range() returns a sequence of numbers
for i in range(3):
    # indentation - 4 spaces recommendation
    # this instruction will be executed
    print("Step number:", i+1)
```

```
Step number: 1
Step number: 2
Step number: 3
```

Review – Python Collections

Collections allow many values in a single variable

- Lists
  - ordered and changeable, allow duplicate
- Sets
  - unordered, unindexed, unchangeable (but you can add and remove items), no duplicate members
- Tuples
  - ordered and unchangeable (immutable), allow duplicate
- Dictionaries
  - Ordered, changeable (in Python 3.7 and later), no duplicate members

Overview – Data Structures (Lists and Sets)

- List Indexing
- How long is a list?
- Concatenating and Slicing Lists
- List Methods
- List Mutability
- Sets Introduction
- Modifying a set in Python

List Introduction

- Lists are created using square brackets [ ]
- Lists can store many elements, separated by commas

Example:

```
# list of integers
num_list = [1, 2, 3]
# list of strings
prog_languages = ['Python', 'Java', 'JavaScript', 'C#', 'C++']
# A list may be of different types, even another list
my_list = [1, "PI", 3.14, [6, 7]]
```

List Introduction (cont.)

- A list can be empty

Example:

```
# empty list
my_list = []
# to check if a list is empty
if my_list:
    print("Not empty list", my_list)
else:
    print("Yes, empty list: ", my_list)
```

```
Yes, empty list:  []
```

List Indexing

- The index() method returns the index of the specified element in the list. The list starts with 0 in Python.

Python	Java	JavaScript	C#	C++
0	1	2	3	4

```
# list of programming languages
prog_languages = ['Python', 'Java', 'JavaScript', 'C#', 'C++']
# find the index of 'Python'
index = prog_languages.index('Python’)
print(index)
```

```
0
```

Is Item in a List?

- **in** operators checks if an element is present in the list
- It returns True of False

Example:

```
# initialize list
numbers = [5, 10, -4, 2, -9, 9, 15]
print(9 in numbers)
print(-2 in numbers)
print(2 not in numbers)
```

```
True
False
False
```

How Long is a List?

- The len() function returns the number of elements in the list
- You can loop through the list using a for loop

Example:

```
# empty list
my_list = []
print("Lenght is:", len(my_list))
my_list = [1, 2, 7, 16]
print("New lenght is:", len(my_list))
# use the range() function to loop through the index number
for i in range(len(my_list)):
    # print all items via index number
    print(my_list[i])
```

```
Lenght is: 0
New lenght is: 4
1
2
7
16
```

Concatenating Lists

- You can create a new list by adding two existing lists together
- You can use the + operator to combine them

Example:

```
# initializing lists
list1 = [1, 2, 3]
list2 = [2, 4, 5, 6]
# concatenation using + operator
list3 = list1 + list2
# list allows duplicates
print(list3)
```

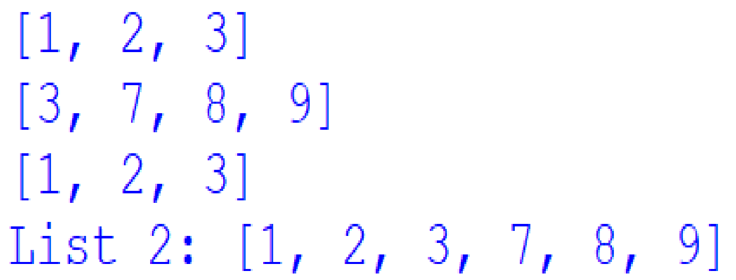
```
[1, 2, 3, 2, 4, 5, 6]
```

Slicing Lists

- Just like string, lists can be sliced, syntax: list[start:stop]

Example:

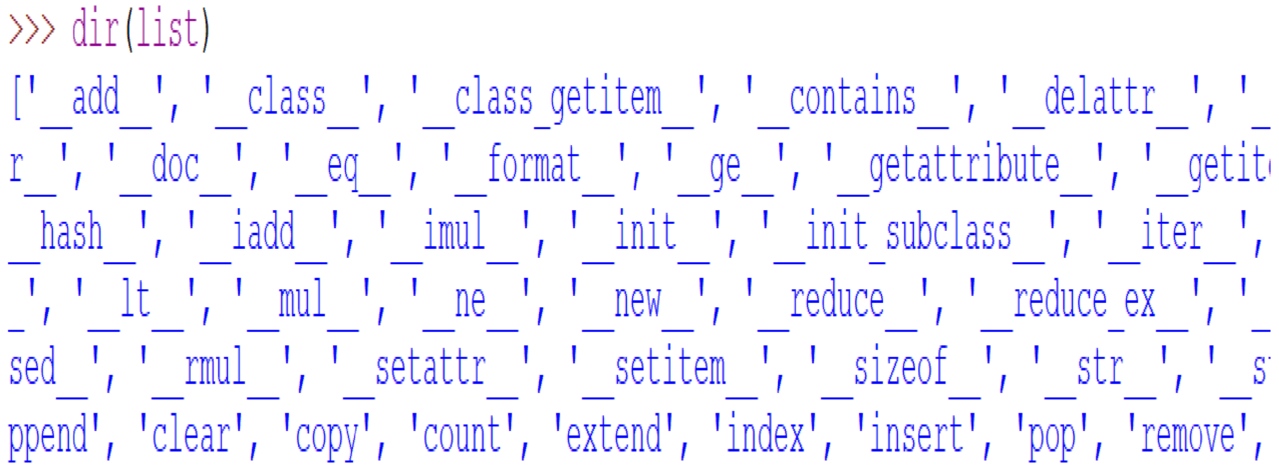
```python
# initializing list
list1 = [1, 2, 3, 7, 8, 9]
print(list1[0:3]) # start through stop-1
print(list1[2:])   # from start to the rest of the list
print(list1[:3])   # from the beginning through stop-1
list2 = list1[:]      # copy the whole list
print("List 2:", list2)
```



```
[1, 2, 3]
[3, 7, 8, 9]
[1, 2, 3]
List 2: [1, 2, 3, 7, 8, 9]
```

Some List Methods

- The **append ()** method – adds an item to the end of the list
- The **insert ()** method – inserts an item at a specified index
- The **clear()** method – removes all items form the list
- The **remove ()** method – removes item form the list
- Python List/Array Methods https://www.w3schools.com/python/python_ref_list.asp

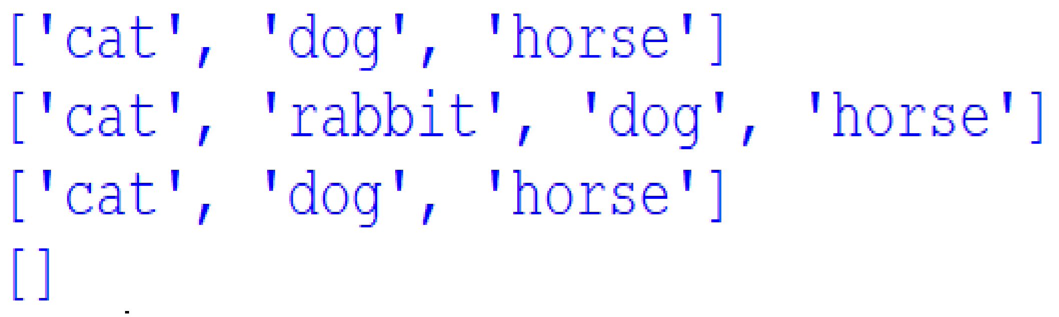


```
>>> dir(list)
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '_
r__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getit
__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__',
_', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '_
sed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__s
ppend', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove',
```

Some List Methods (cont.)

Example:

```
pets = ['cat', 'dog']
pets.append('horse')  # appends the value to the end of the list
print(pets)
pets.insert(1, 'rabbit')  # inserts the value at the position 1
print(pets)
pets.remove('rabbit’)  # removes ‘rabbit’
print(pets)
pets.clear()  # removes all the elements
print(pets)
```



```
['cat', 'dog', 'horse']
['cat', 'rabbit', 'dog', 'horse']
['cat', 'dog', 'horse']
[]
```

Some Build-in Functions and Lists

- There are number of functions built into Python (Built-in Functions, https://docs.python.org/3/library/functions.html) and some can be used with lists.

For example:

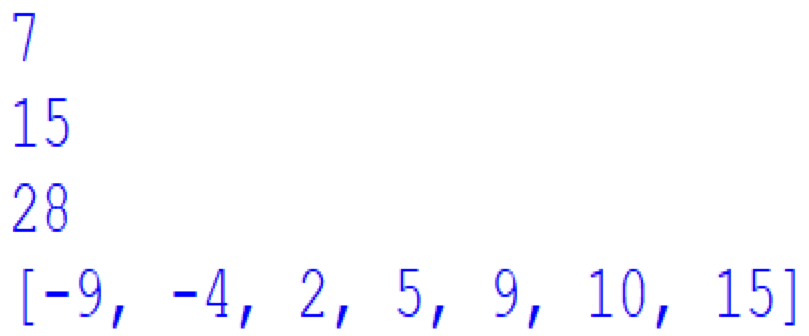
numbers = [5, 10, -4, 2, -9, 9, 15]

print(**len**(numbers))

print(**max**(numbers))

print(**sum**(numbers))

print(**sorted**(numbers))



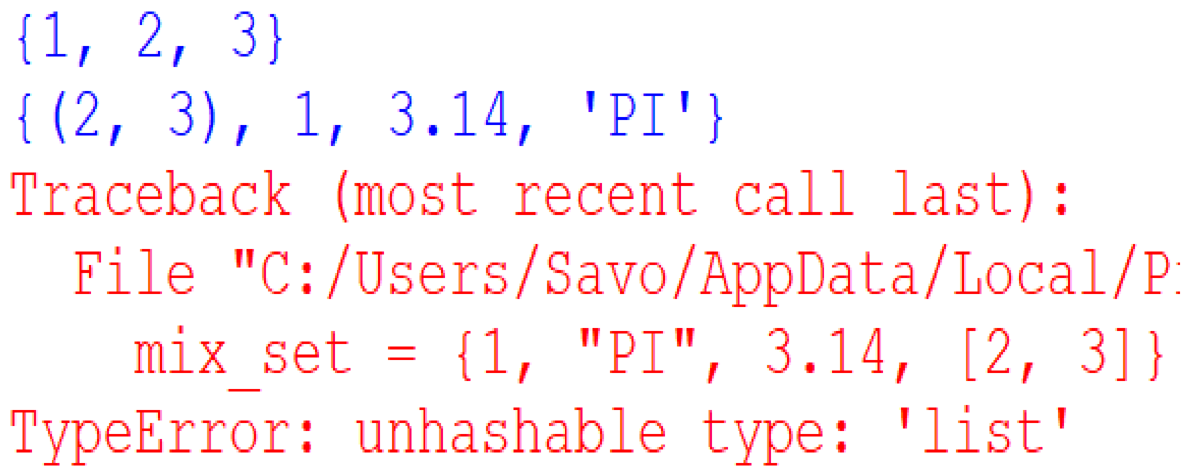
```
7
15
28
[-9, -4, 2, 5, 9, 10, 15]
```

Sets Introduction

- Sets are created by placing the items inside curly braces { }. You cannot be sure in which order the items will appear.

Example:

```
# set of integers
num_set = {1, 2, 3}
print(num_set)
# A set may be of different types
mix_set = {1, "PI", 3.14, (2, 3)}
print(mix_set)  # order is lost?
# it cannot have mutable elements like lists
mix_set = {1, "PI", 3.14, [2, 3]}
```



```
{1, 2, 3}
{(2, 3), 1, 3.14, 'PI'}
Traceback (most recent call last):
  File "C:/Users/Savo/AppData/Local/P
    mix_set = {1, "PI", 3.14, [2, 3]}
TypeError: unhashable type: 'list'
```

Modifying a set in Python

- Since sets are unordered, indexing has no meaning
- Although sets are **immutable**, you can add a single element using the add() or update () method.

Example:

```
# initialize set
num_set = {1, 2, 3, 4}
print(num_set)
# add multiple elements
num_set.update({5, 6})
print(num_set)
num_set.add(7) # add a single element
print(num_set)
```

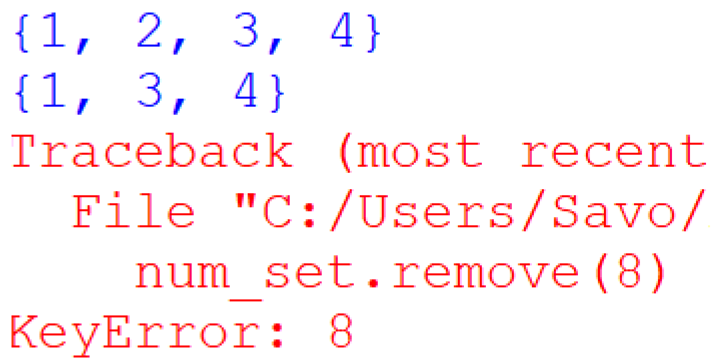
```
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6, 7}
```

Modifying a set in Python (cont.)

- You can remove an element using remove() or discard() method
- The discard() leaves a set unchanged if element is missing
- The remove() will raise an error if element is not present

Example:

```
# intialize set
num_set = {1, 2, 3, 4}
print(num_set)
num_set.discard(2)   #discard an element
print(num_set)
num_set.discard(8)  # will not raise an error
num_set.remove(8)  # KeyError, 8 is not present
```



```
{1, 2, 3, 4}
{1, 3, 4}
Traceback (most recent
  File "C:/Users/Savo/
    num_set.remove(8)
KeyError: 8
```

Questions?