Introduction to Software Development

lecture: session 6

level 3

University of West London, UK

Christian Sauer

Flow Control Structures

Selection

The if statement is used to check a condition: *if* the condition is true, we run a block of statements, *elif*, *else* we process another block(s) of statements

if … **elif** … **else**

Making choices

e.g. Menu options

Repetition

Repeat a set of statements under some conditions

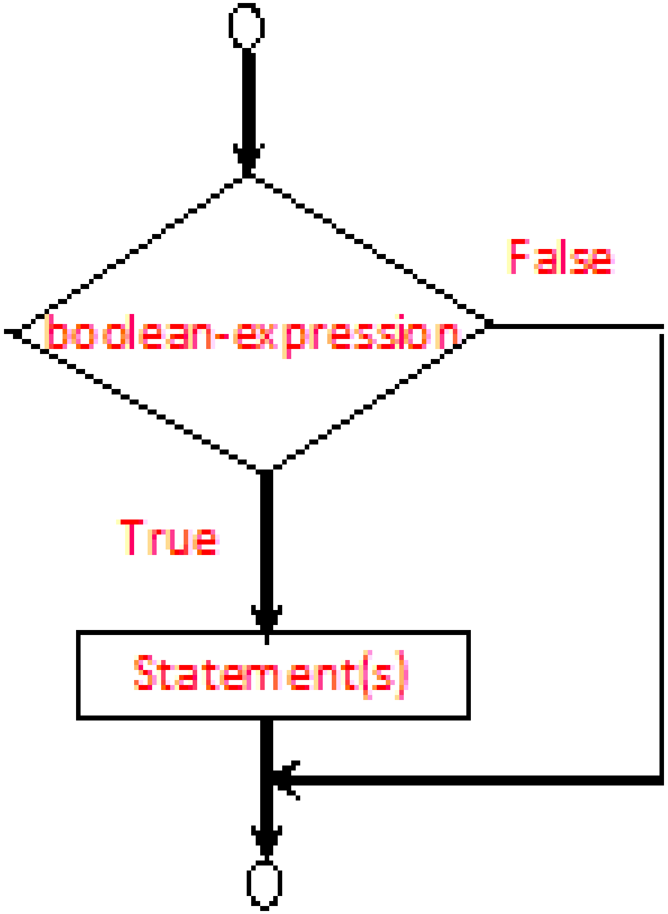
The **while** statement.
It repeats a set of statements while some condition is True.

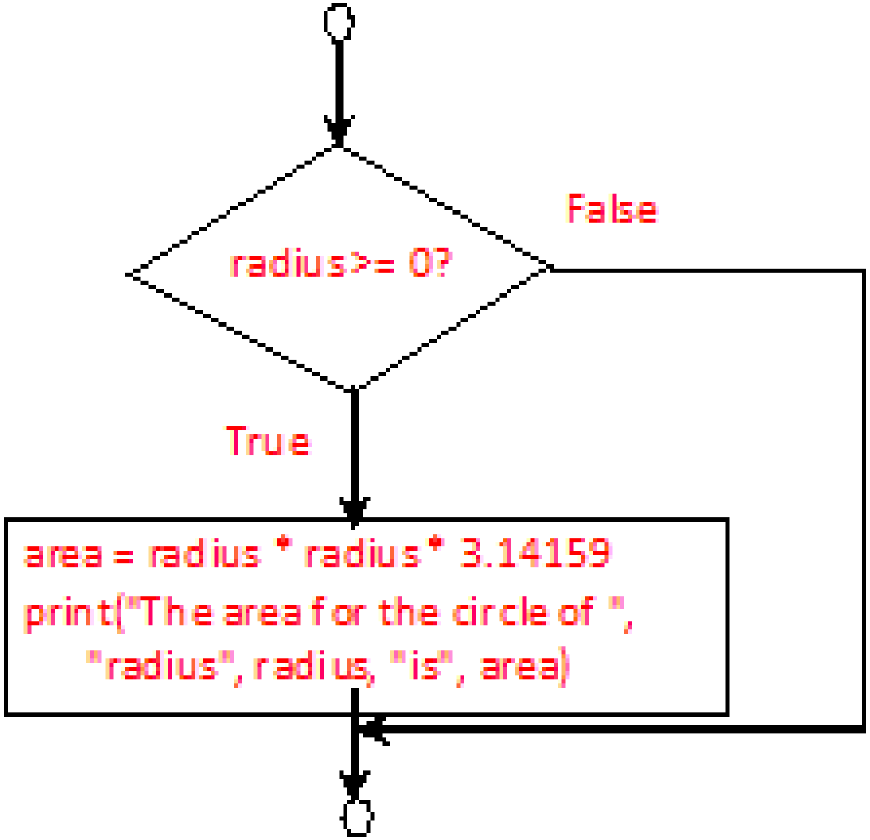
The **for** statement.
Repeats statements for a set number of times.
Iterates over a sequence of objects

One-way IF Statements

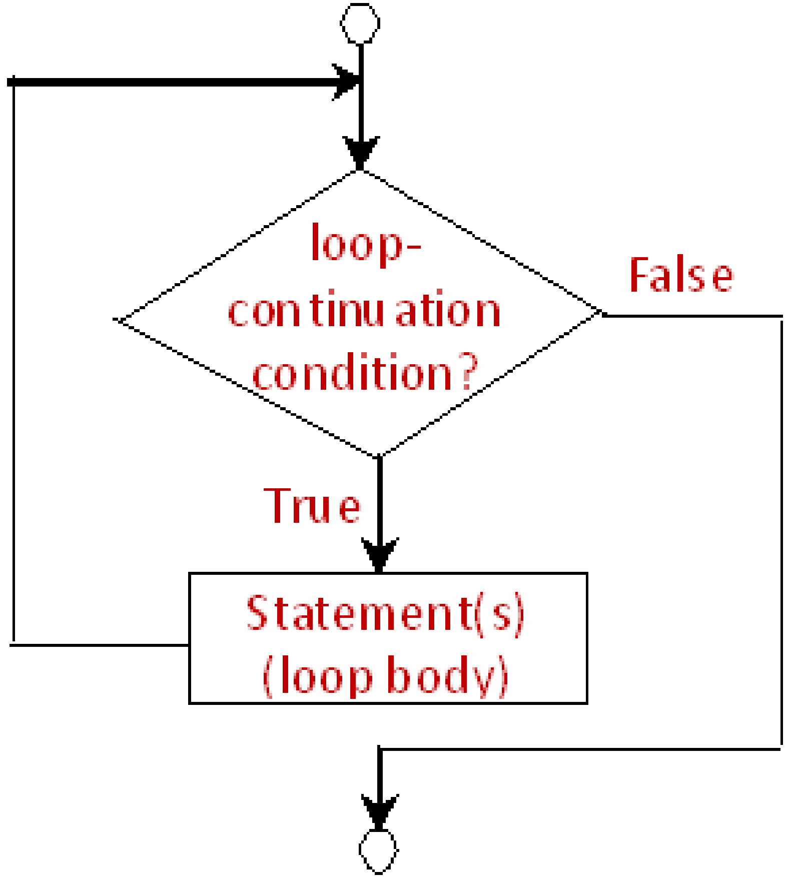
```
if boolean-expression:
    statement(s)
```

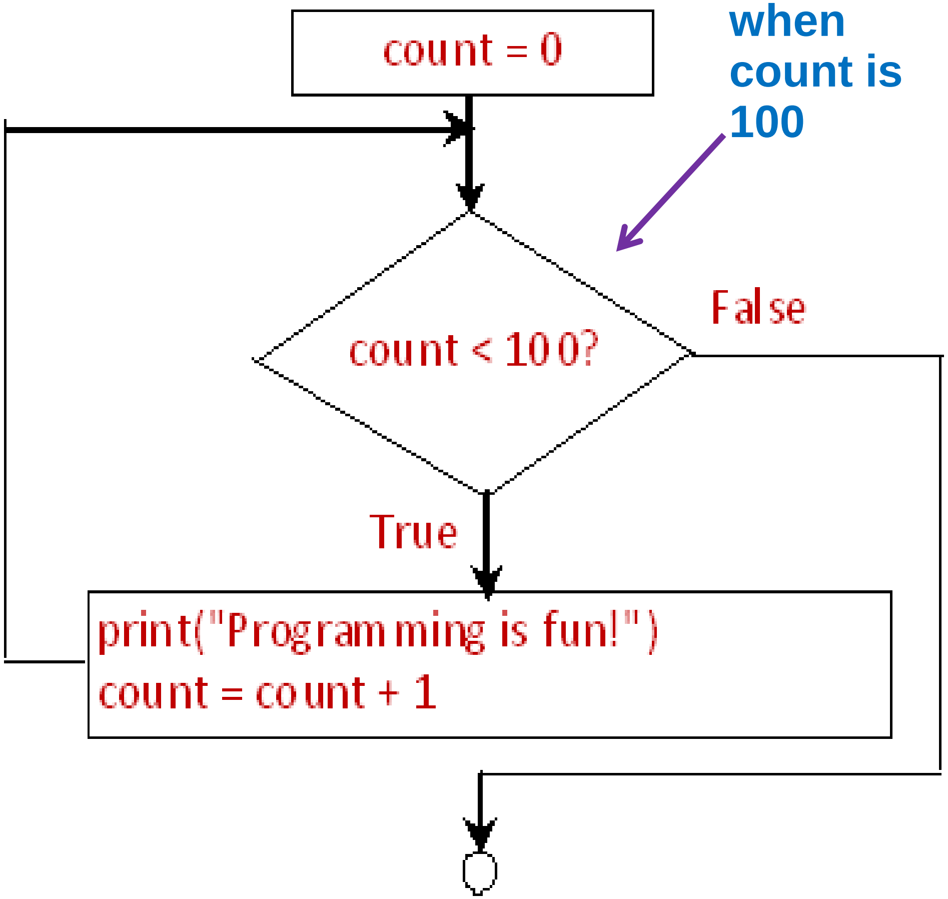
```
if radius >= 0:
    area = radius * radius * 3.14159
    print("The area for the circle of radius“,
          radius, "is“, area)
```




False
radius>= 0?
True
area = radius * radius * 3.14159
print("The area for the circle of ",
"radius", radius, "is", area)

While Loop: Flow Chart

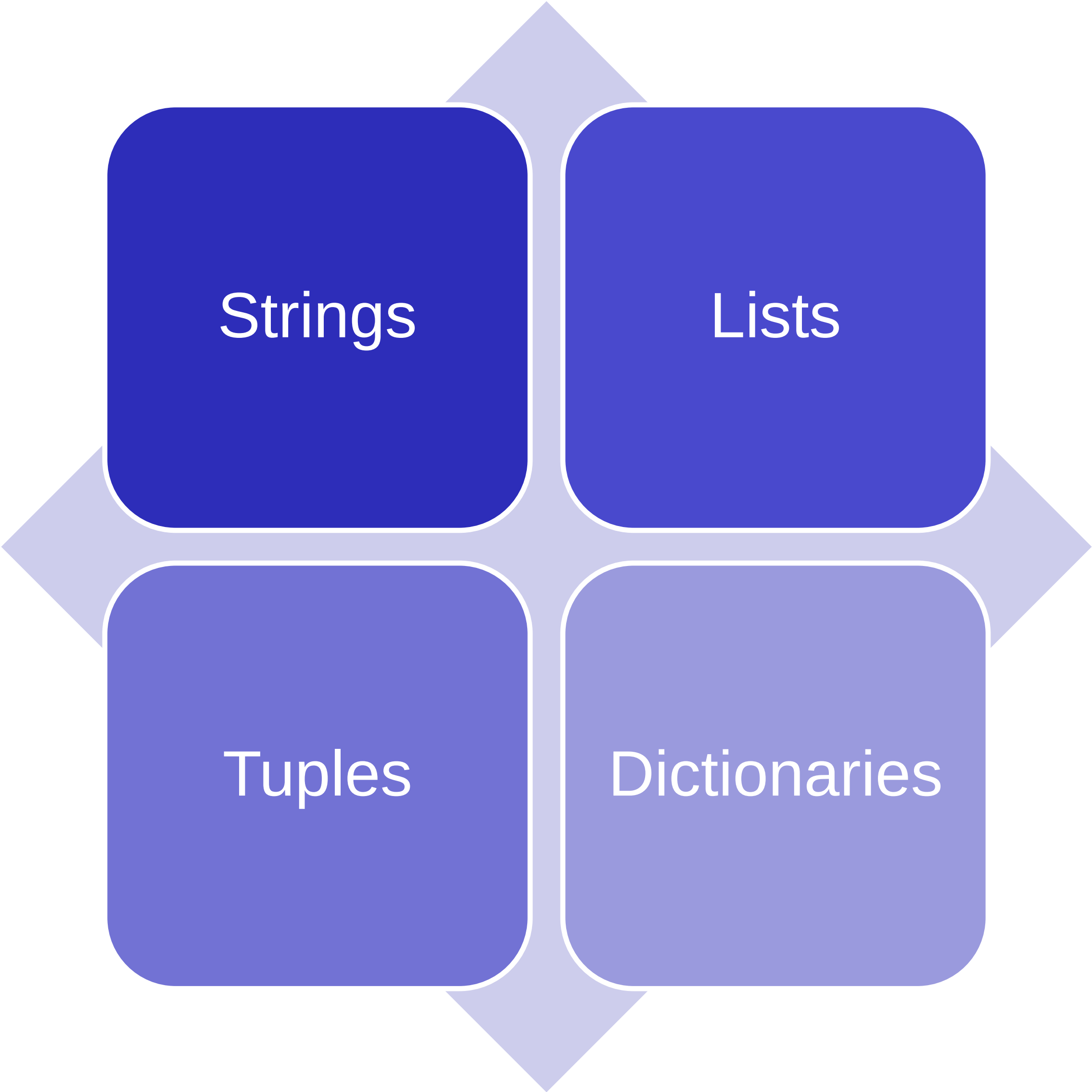




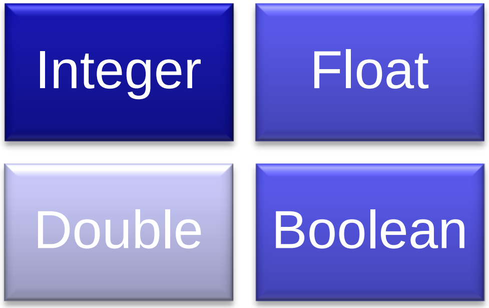
Loop ends when count is 100

Data Structures

Data structures that contain more than one item!



Data Structures

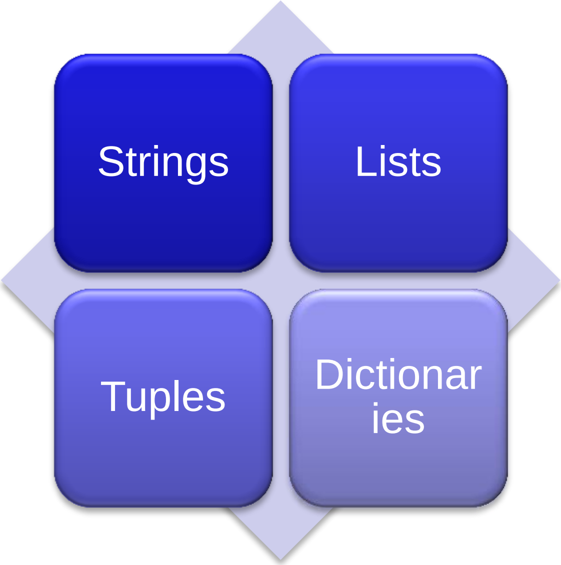


Integer

Float

Double

Boolean

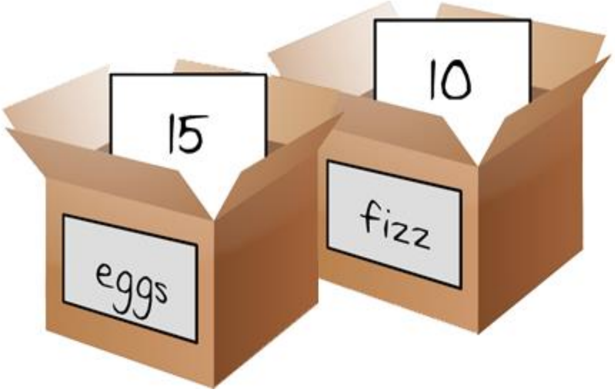


Strings

Lists

Tuples

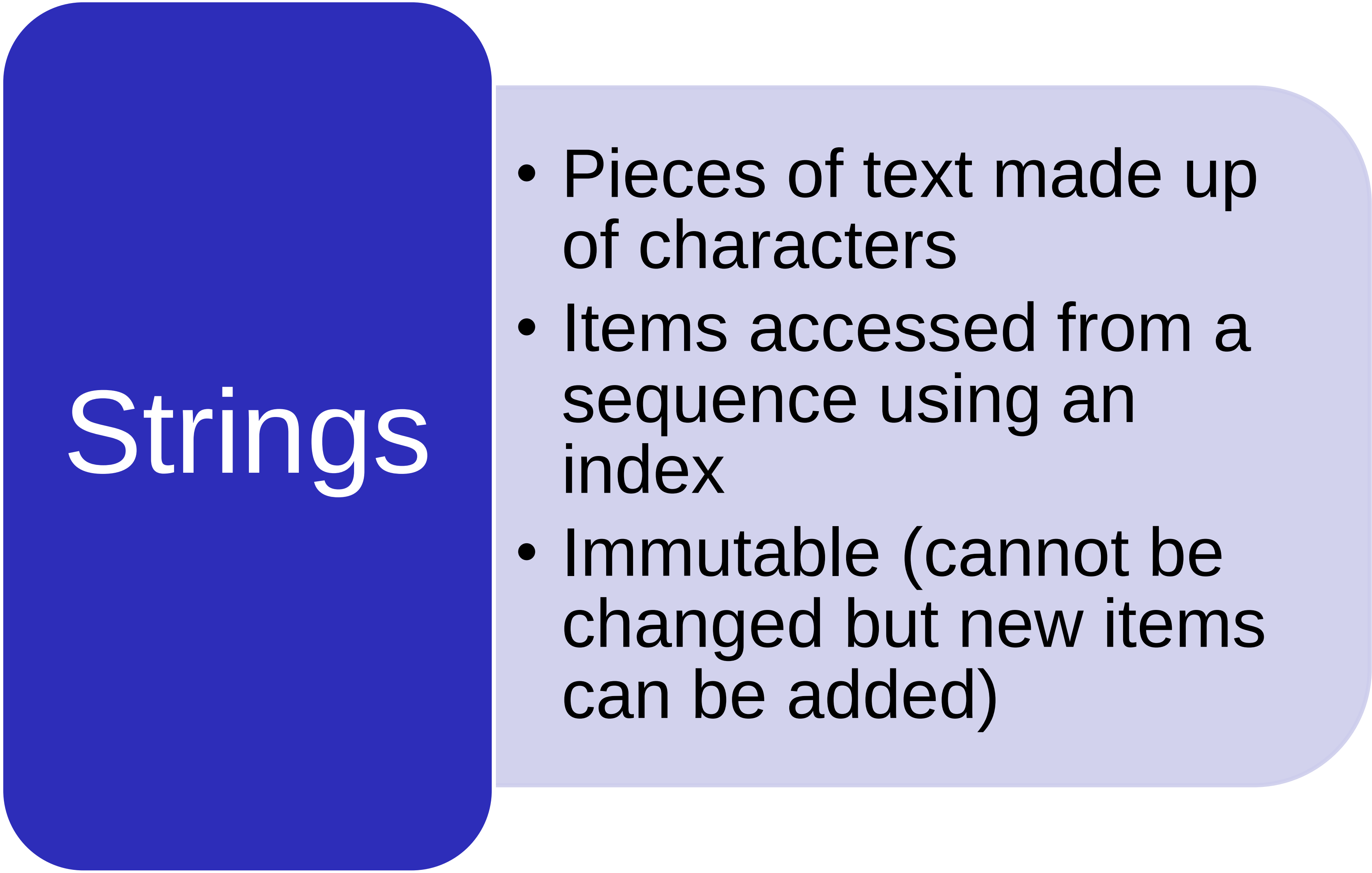
Dictionaries



Variables can only contain one value

Data structures can contain more than one item!

Data Structures: Main Differences

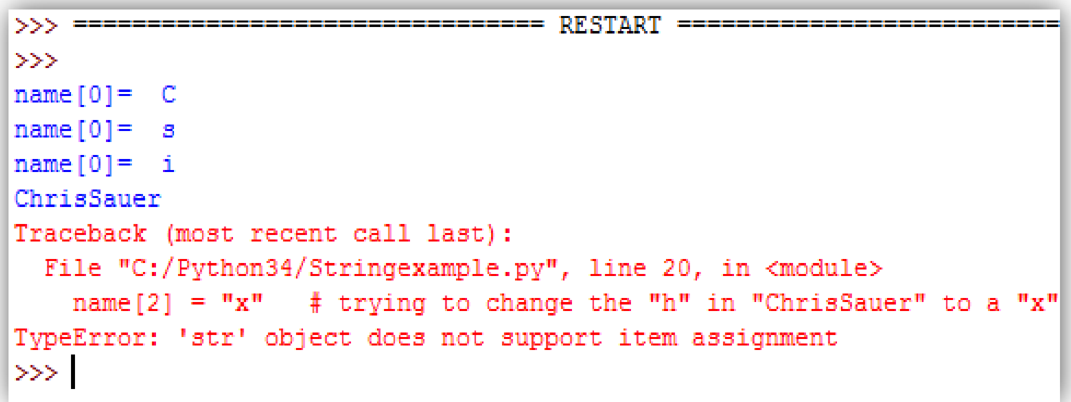


Strings

- Pieces of text made up of characters
- Items accessed from a sequence using an index
- Immutable (cannot be changed but new items can be added)

Data Structures: Strings

```
# Demonstrate properties of String
# Create a string

name = "Chris"

# Access characters in string by using an index

print("name[0]= ",name[0])
print("name[0]= ",name[4])
print("name[0]= ",name[-2])

# Adding new characters to the string

surname = "Sauer"
name = name + surname
print(name)

# Trying to mutate (change) existing characters
# trying to change "h" in "ChrisSauer" to "x"
# This will go wrong...

name[2] = "x"
```



```
>>> ================================ RESTART ==========================
>>>
name[0]=  C
name[0]=  s
name[0]=  i
ChrisSauer
Traceback (most recent call last):
  File "C:/Python34/Stringexample.py", line 20, in <module>
    name[2] = "x"   # trying to change the "h" in "ChrisSauer" to a "x"
TypeError: 'str' object does not support item assignment
>>> |
```

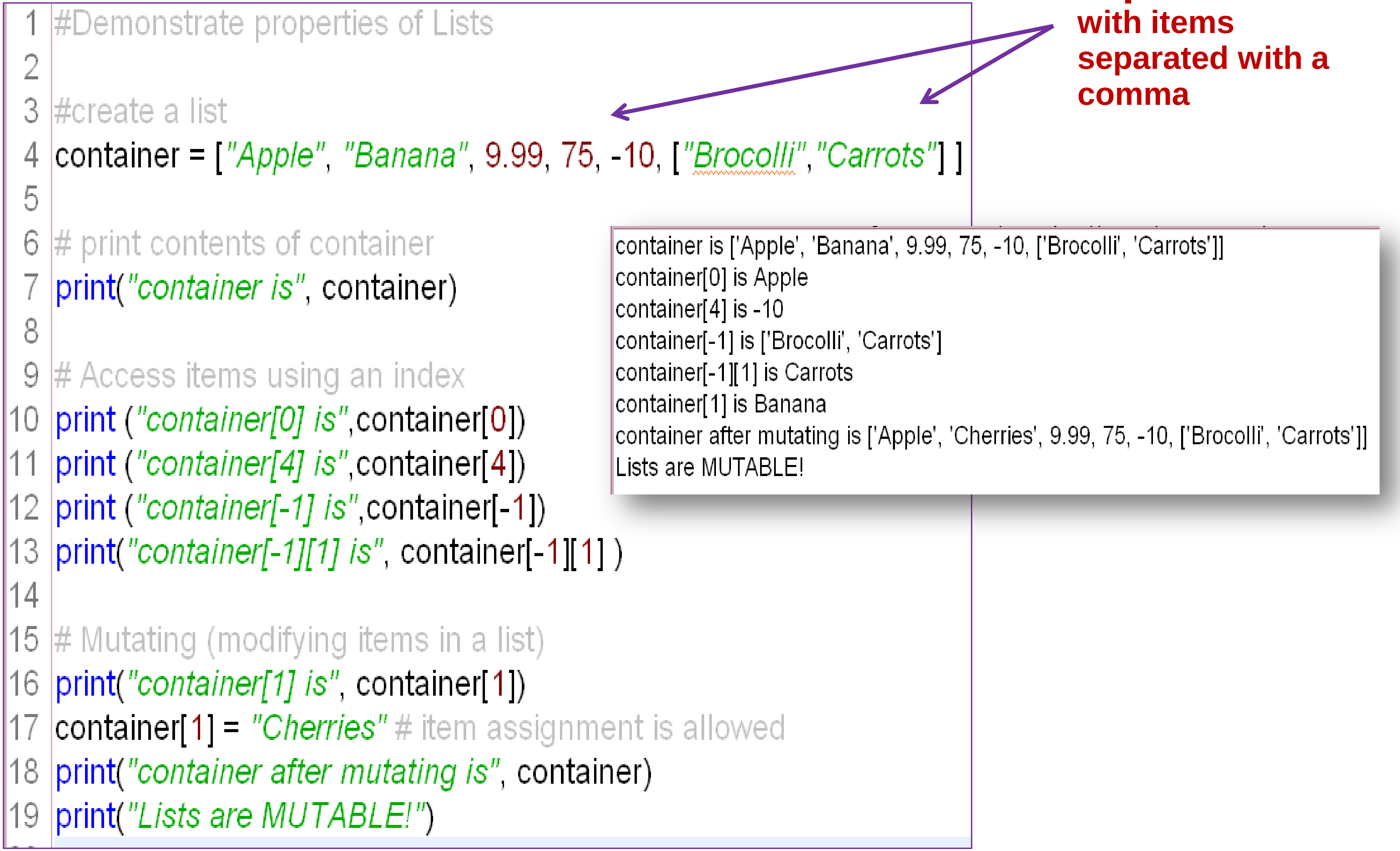
Data Structures: Main Differences

Lists

- Ordered groups of individual data items (integer, float, sub-list, string etc)
- Items accessed from a sequence using an index
- Mutable (can be modified; items can be altered, added or removed)

Data Structures: Lists

Square brackets with items separated with a comma



```python
#Demonstrate properties of Lists

#create a list
container = ["Apple", "Banana", 9.99, 75, -10, ["Brocolli","Carrots"] ]

# print contents of container
print("container is", container)

# Access items using an index
print ("container[0] is",container[0])
print ("container[4] is",container[4])
print ("container[-1] is",container[-1])
print("container[-1][1] is", container[-1][1] )

# Mutating (modifying items in a list)
print("container[1] is", container[1])
container[1] = "Cherries" # item assignment is allowed
print("container after mutating is", container)
print("Lists are MUTABLE!")
```

```
container is ['Apple', 'Banana', 9.99, 75, -10, ['Brocolli', 'Carrots']]
container[0] is Apple
container[4] is -10
container[-1] is ['Brocolli', 'Carrots']
container[-1][1] is Carrots
container[1] is Banana
container after mutating is ['Apple', 'Cherries', 9.99, 75, -10, ['Brocolli', 'Carrots']]
Lists are MUTABLE!
```

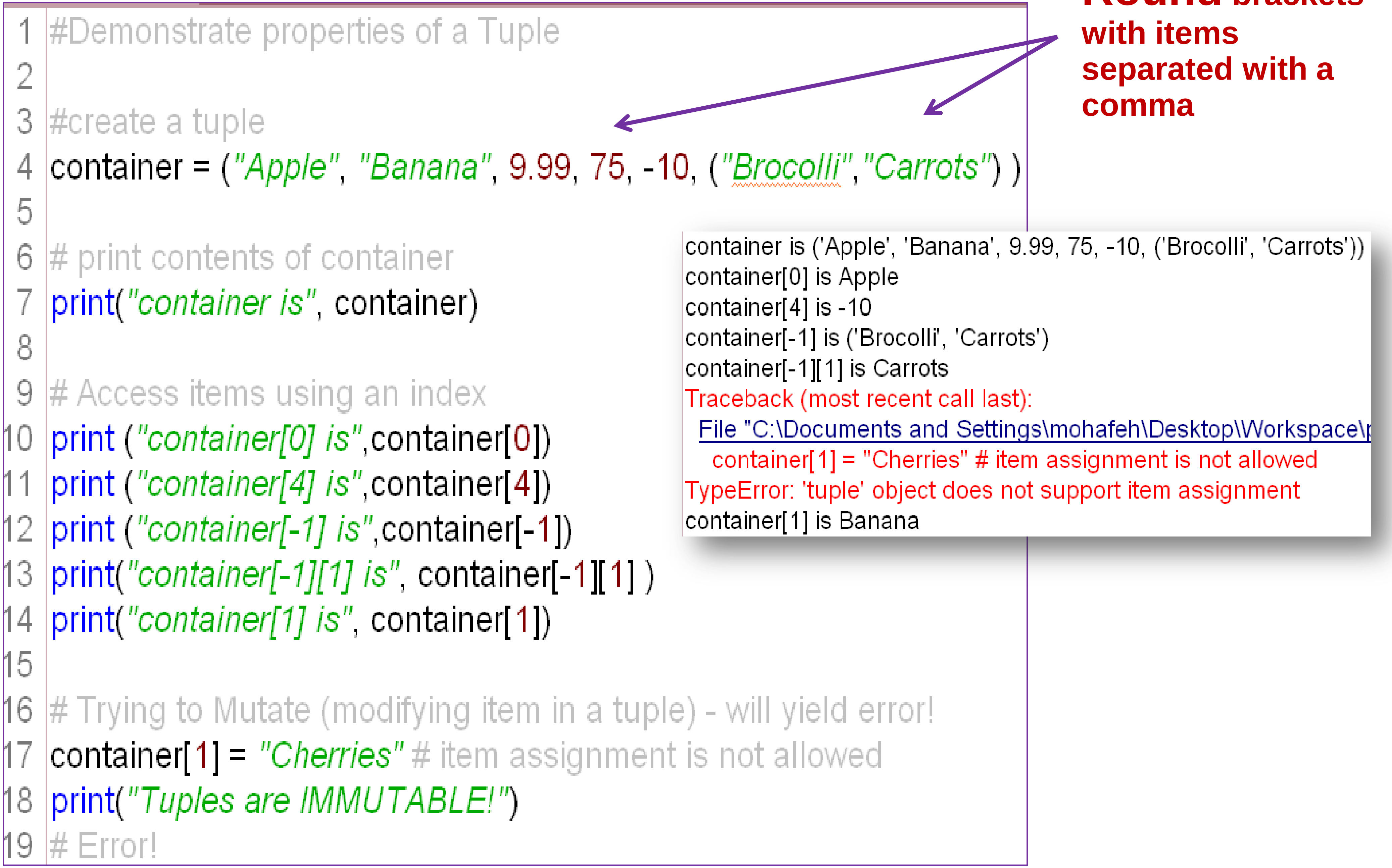
Data Structures: Main Differences

Tuples

- Ordered groups of individual data items
- Items accessed from a sequence using an index
- Immutable (cannot be changed but new items can be added)
- Sealed packets of information. Useful in situations where a set of values has to be passed on to another place securely. Also used to provide dictionary keys.

Data Structures: Tuples

Round brackets with items separated with a comma



```python
#Demonstrate properties of a Tuple

#create a tuple
container = ("Apple", "Banana", 9.99, 75, -10, ("Brocolli","Carrots") )

# print contents of container
print("container is", container)

# Access items using an index
print ("container[0] is",container[0])
print ("container[4] is",container[4])
print ("container[-1] is",container[-1])
print("container[-1][1] is", container[-1][1] )
print("container[1] is", container[1])

# Trying to Mutate (modifying item in a tuple) - will yield error!
container[1] = "Cherries" # item assignment is not allowed
print("Tuples are IMMUTABLE!")
# Error!
```

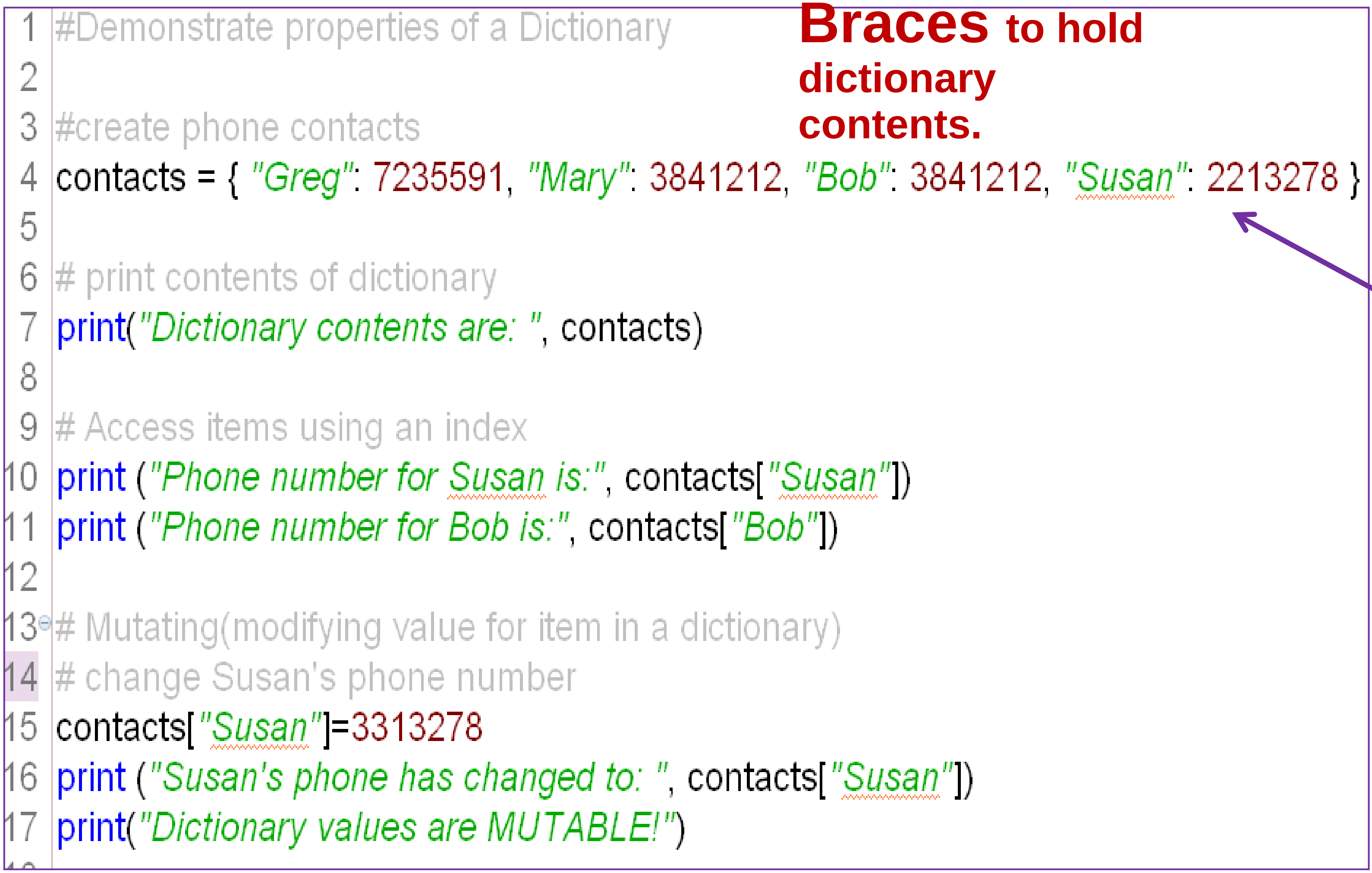
```
container is ('Apple', 'Banana', 9.99, 75, -10, ('Brocolli', 'Carrots'))
container[0] is Apple
container[4] is -10
container[-1] is ('Brocolli', 'Carrots')
container[-1][1] is Carrots
Traceback (most recent call last):
  File "C:\Documents and Settings\mohafeh\Desktop\Workspace\p
    container[1] = "Cherries" # item assignment is not allowed
TypeError: 'tuple' object does not support item assignment
container[1] is Banana
```

Data Structures: Main Differences

Dictionaries

- Groups of key-value pairs
- Dictionary itself is a mutable data type, which means you can add, remove and modify key-value pairs. The keys are said to be mapped to the assigned values.

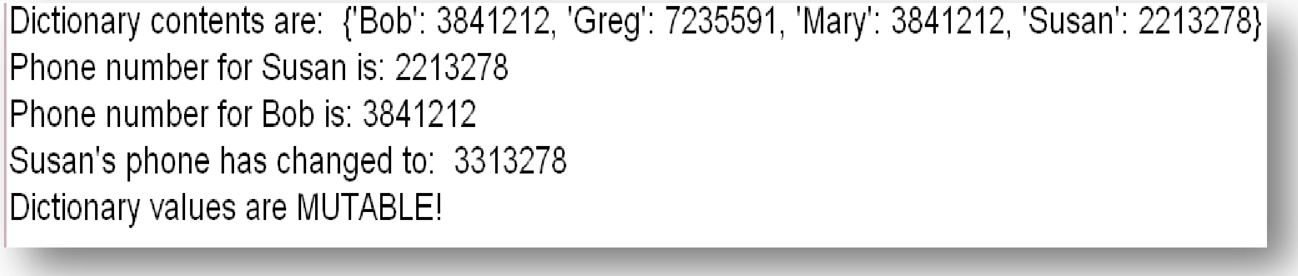
Data Structures: Dictionaries



```
#Demonstrate properties of a Dictionary

#create phone contacts
contacts = { "Greg": 7235591, "Mary": 3841212, "Bob": 3841212, "Susan": 2213278 }

# print contents of dictionary
print("Dictionary contents are: ", contacts)

# Access items using an index
print ("Phone number for Susan is:", contacts["Susan"])
print ("Phone number for Bob is:", contacts["Bob"])

# Mutating(modifying value for item in a dictionary)
# change Susan's phone number
contacts["Susan"]=3313278
print ("Susan's phone has changed to: ", contacts["Susan"])
print("Dictionary values are MUTABLE!")
```

Braces to hold dictionary contents.

Key and Value separated with a colon



```
Dictionary contents are:  {'Bob': 3841212, 'Greg': 7235591, 'Mary': 3841212, 'Susan': 2213278}
Phone number for Susan is: 2213278
Phone number for Bob is: 3841212
Susan's phone has changed to:  3313278
Dictionary values are MUTABLE!
```

What operations can we perform within a dictionary?

- Access an item
- Determine length
- Add an item
  - Append
  - Insert
- Change an item
- Delete an item
- Find an item
- Sort items or values
- Iterate or traverse through items or values

Common Operations on Data Structures

	String	List	Tuple	Dictionary
Access an item	✓	✓	✓	✓
Determine length	✓	✓	✓	✓
Append an item to the end	✗	✓	✗	✓
Insert an item to given position	✗	✓	✗	✗
Delete an item	✗	✓	✗	✓
Change an item	✗	✓	✗	✓
Find an item	✓	✓	✓	✓
Sort items (same data type)	✗	✓	✗	✗
Iterate or traverse	✓	✓	✓	✓

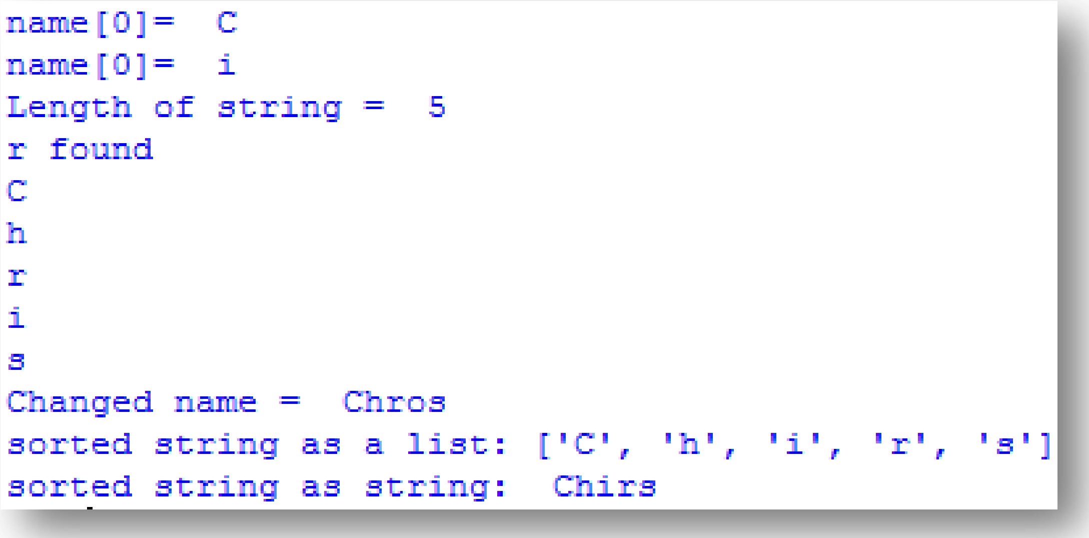
	String name	List container	Tuple container	Dictionary contacts
Access an item	name[2]	container[0]	container[-1]	contacts[key]
Determine length	len(name)	len(container)	len(container)	len(contacts)
Append an item to the end	X	container.append(item)	X	contacts[key]=value
Insert an item to given position	X	container.insert(index, item)	X	X

Delete an item	X	container.remove(item)
		container.pop()
		container.pop(index)
			X	contacts.pop(key)

Change an item	X	re-assign using index	X	re-assign using key
Find an item	if letter in name:	if item in container:	if item in container:	if key in contacts:
Sort items	sorted(name)	container.sort()	X	X
(same data type)	“”.join(sorted(name))			
Iterate or traverse	for letter in name:	for item in container:	for item in container:	for key in contacts:

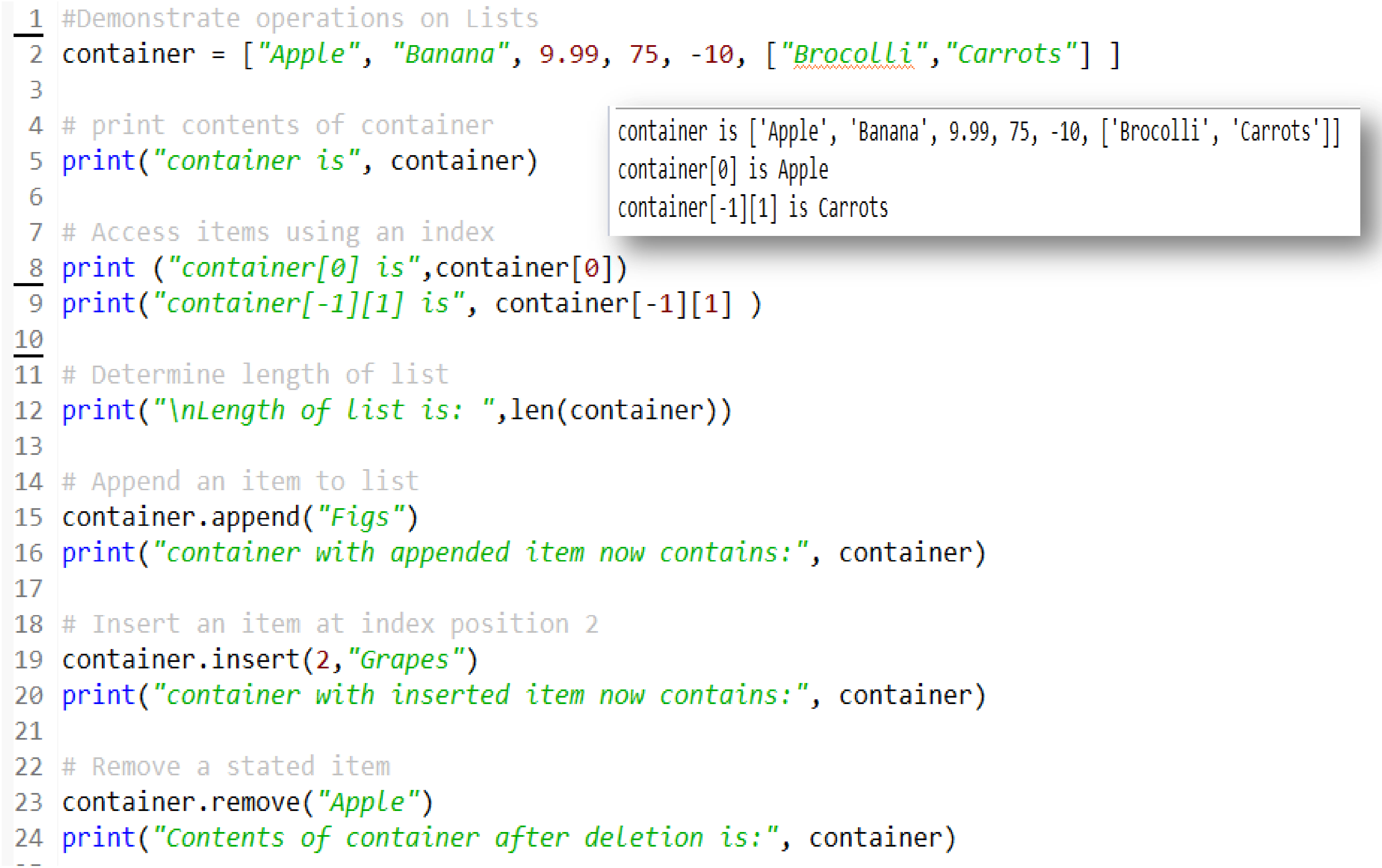
String Operations

```
# Demonstrate operations on String

# Create a string
name = "Chris"

# Access characters in string by using an index
print("name[0]= ",name[0])
print("name[0]= ",name[-2])

# Determine the length of a string
print("Length of string = ", len(name))

# Find a letter in name
if "r" in name:
    print("r found")

# Iterate / Traverse a string
for letter in name:
    print(letter)

#Deleting a character from a string
#TRICK: as strings are imutable
#so we can't change characters, therefore
#we replace the character with an empty space
new_name = name.replace("i", "o")
print("Changed name = ",new_name)

#Sort letters in a string
print("sorted string as a list:", sorted(name))


#Sort string and display as joined string
print("sorted string as string: ", "".join(sorted(name)))
```



```
name[0]=  C
name[0]=  i
Length of string =  5
r found
C
h
r
i
s
Changed name =  Chros
sorted string as a list: ['C', 'h', 'i', 'r', 's']
sorted string as string:  Chirs
```

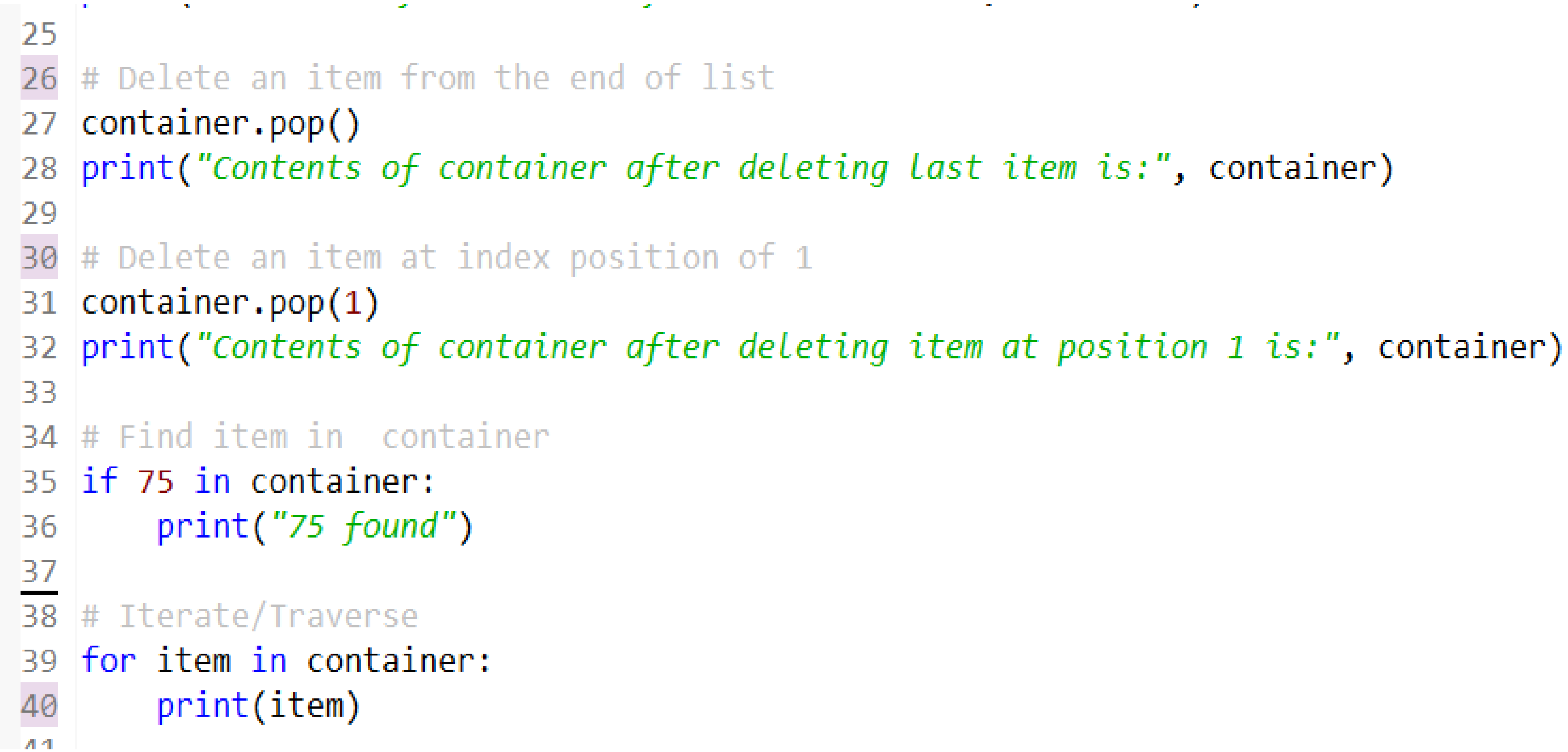
List Operations



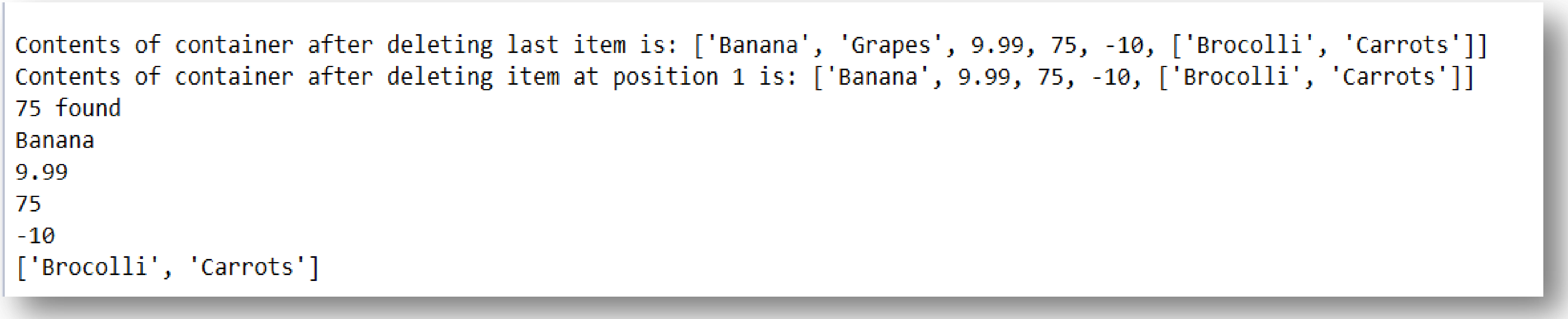
```
#Demonstrate operations on Lists
container = ["Apple", "Banana", 9.99, 75, -10, ["Brocolli","Carrots"] ]

# print contents of container
print("container is", container)

# Access items using an index
print ("container[0] is",container[0])
print("container[-1][1] is", container[-1][1] )

# Determine length of list
print("\nLength of list is: ",len(container))

# Append an item to list
container.append("Figs")
print("container with appended item now contains:", container)

# Insert an item at index position 2
container.insert(2,"Grapes")
print("container with inserted item now contains:", container)

# Remove a stated item
container.remove("Apple")
print("Contents of container after deletion is:", container)
```

```
container is ['Apple', 'Banana', 9.99, 75, -10, ['Brocolli', 'Carrots']]
container[0] is Apple
container[-1][1] is Carrots
```

List Operations

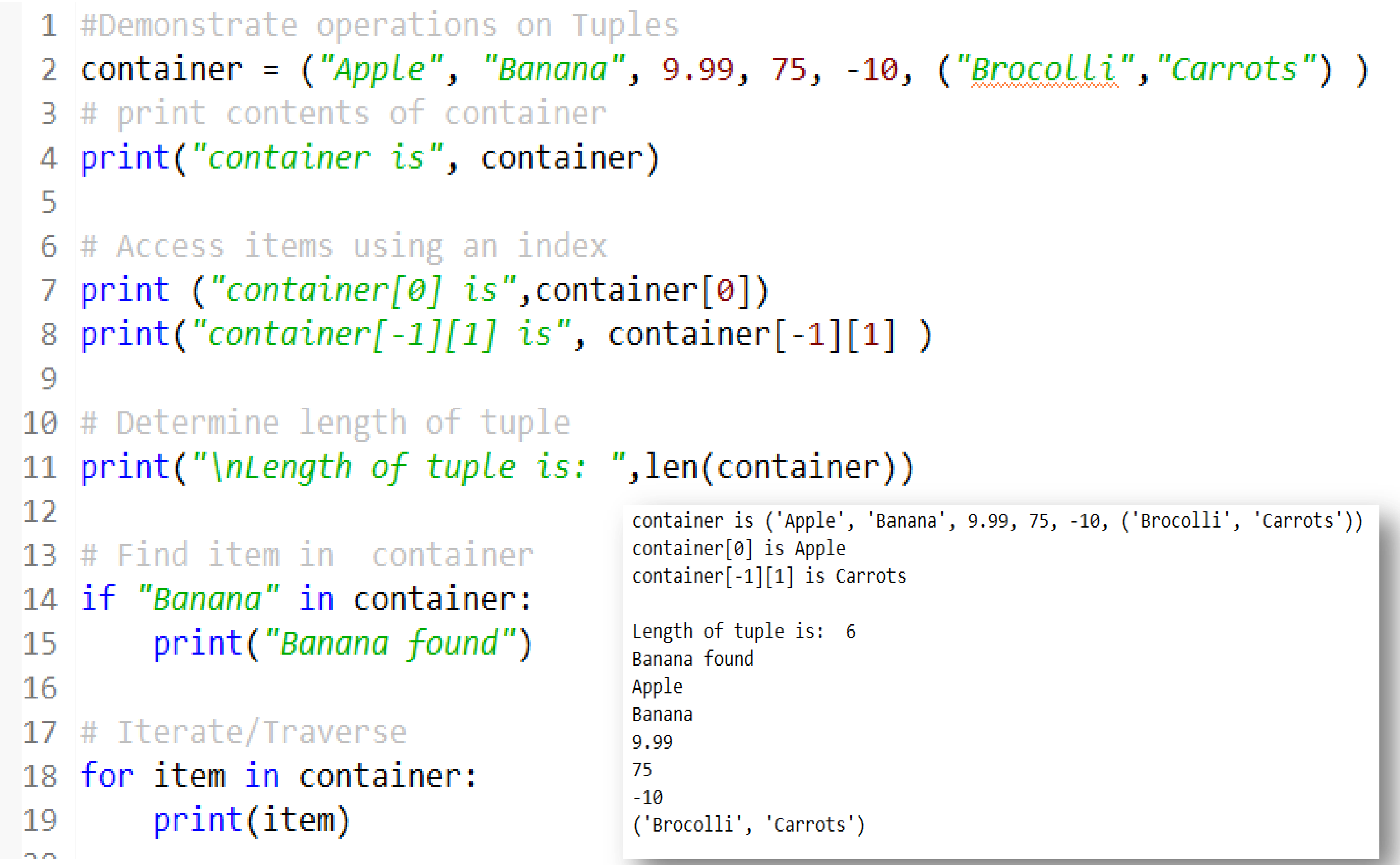


```python

# Delete an item from the end of list
container.pop()
print("Contents of container after deleting last item is:", container)

# Delete an item at index position of 1
container.pop(1)
print("Contents of container after deleting item at position 1 is:", container)

# Find item in  container
if 75 in container:
    print("75 found")

# Iterate/Traverse
for item in container:
    print(item)
```



```
Contents of container after deleting last item is: ['Banana', 'Grapes', 9.99, 75, -10, ['Brocolli', 'Carrots']]
Contents of container after deleting item at position 1 is: ['Banana', 9.99, 75, -10, ['Brocolli', 'Carrots']]
75 found
Banana
9.99
75
-10
['Brocolli', 'Carrots']
```

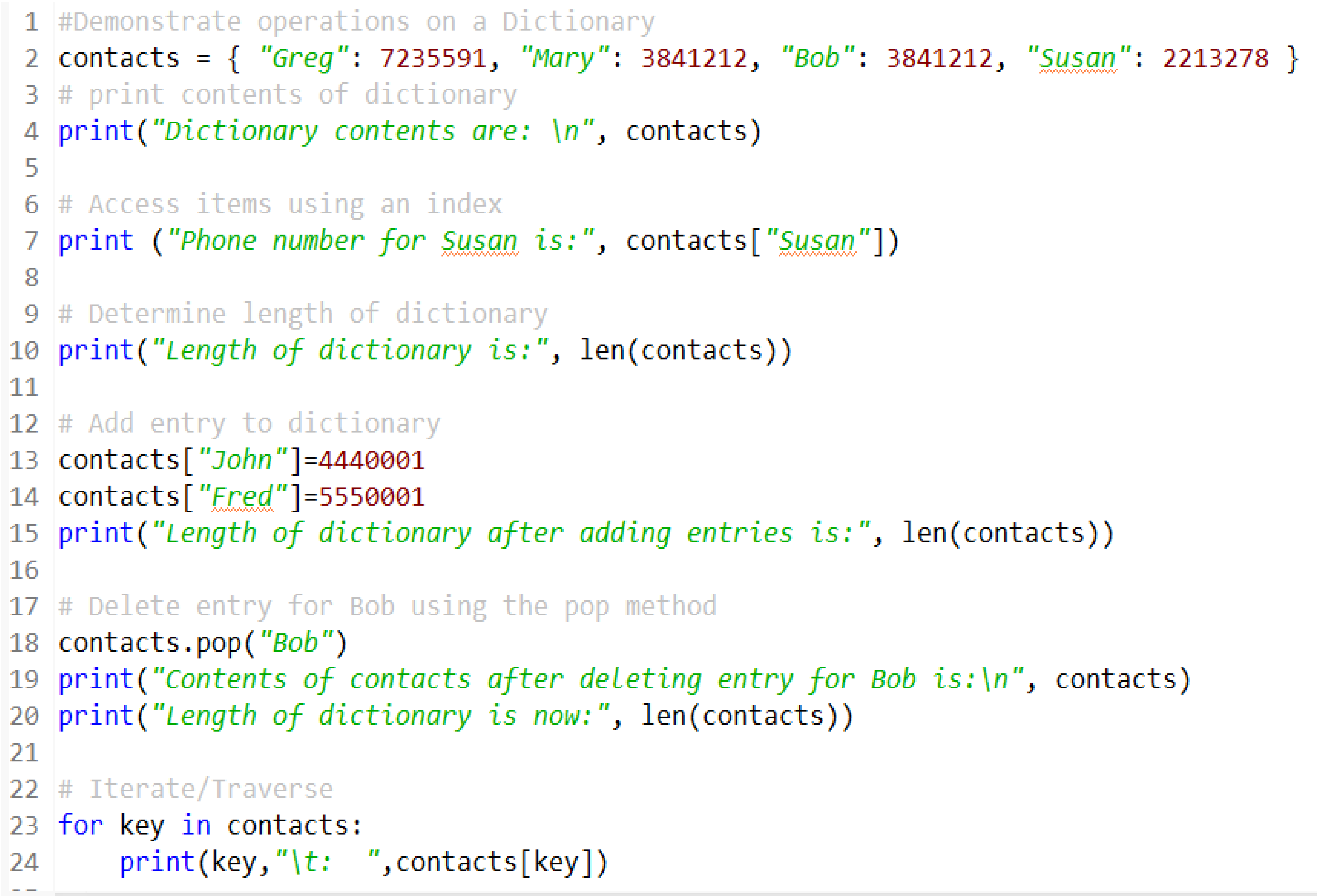
Tuple Operations



```python
#Demonstrate operations on Tuples
container = ("Apple", "Banana", 9.99, 75, -10, ("Brocolli","Carrots") )
# print contents of container
print("container is", container)

# Access items using an index
print ("container[0] is",container[0])
print("container[-1][1] is", container[-1][1] )

# Determine length of tuple
print("\nLength of tuple is: ",len(container))

# Find item in  container
if "Banana" in container:
    print("Banana found")

# Iterate/Traverse
for item in container:
    print(item)
```

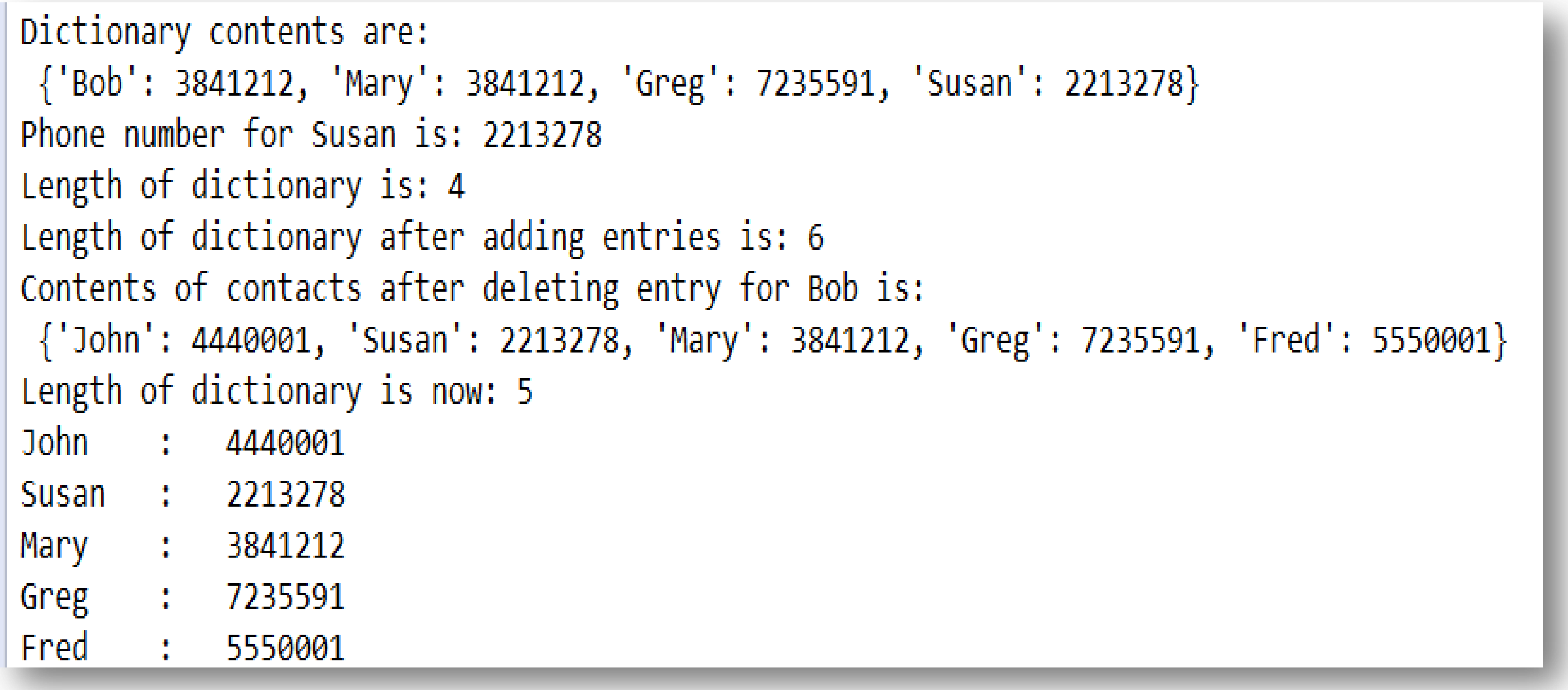
```
container is ('Apple', 'Banana', 9.99, 75, -10, ('Brocolli', 'Carrots'))
container[0] is Apple
container[-1][1] is Carrots

Length of tuple is:  6
Banana found
Apple
Banana
9.99
75
-10
('Brocolli', 'Carrots')
```

Dictionary Operations



```python
#Demonstrate operations on a Dictionary
contacts = { "Greg": 7235591, "Mary": 3841212, "Bob": 3841212, "Susan": 2213278 }
# print contents of dictionary
print("Dictionary contents are: \n", contacts)

# Access items using an index
print ("Phone number for Susan is:", contacts["Susan"])

# Determine length of dictionary
print("Length of dictionary is:", len(contacts))

# Add entry to dictionary
contacts["John"]=4440001
contacts["Fred"]=5550001
print("Length of dictionary after adding entries is:", len(contacts))

# Delete entry for Bob using the pop method
contacts.pop("Bob")
print("Contents of contacts after deleting entry for Bob is:\n", contacts)
print("Length of dictionary is now:", len(contacts))

# Iterate/Traverse
for key in contacts:
    print(key,"\t:  ",contacts[key])
```

Dictionary Operations



```
Dictionary contents are:
 {'Bob': 3841212, 'Mary': 3841212, 'Greg': 7235591, 'Susan': 2213278}
Phone number for Susan is: 2213278
Length of dictionary is: 4
Length of dictionary after adding entries is: 6
Contents of contacts after deleting entry for Bob is:
 {'John': 4440001, 'Susan': 2213278, 'Mary': 3841212, 'Greg': 7235591, 'Fred': 5550001}
Length of dictionary is now: 5
John    :   4440001
Susan   :   2213278
Mary    :   3841212
Greg    :   7235591
Fred    :   5550001
```