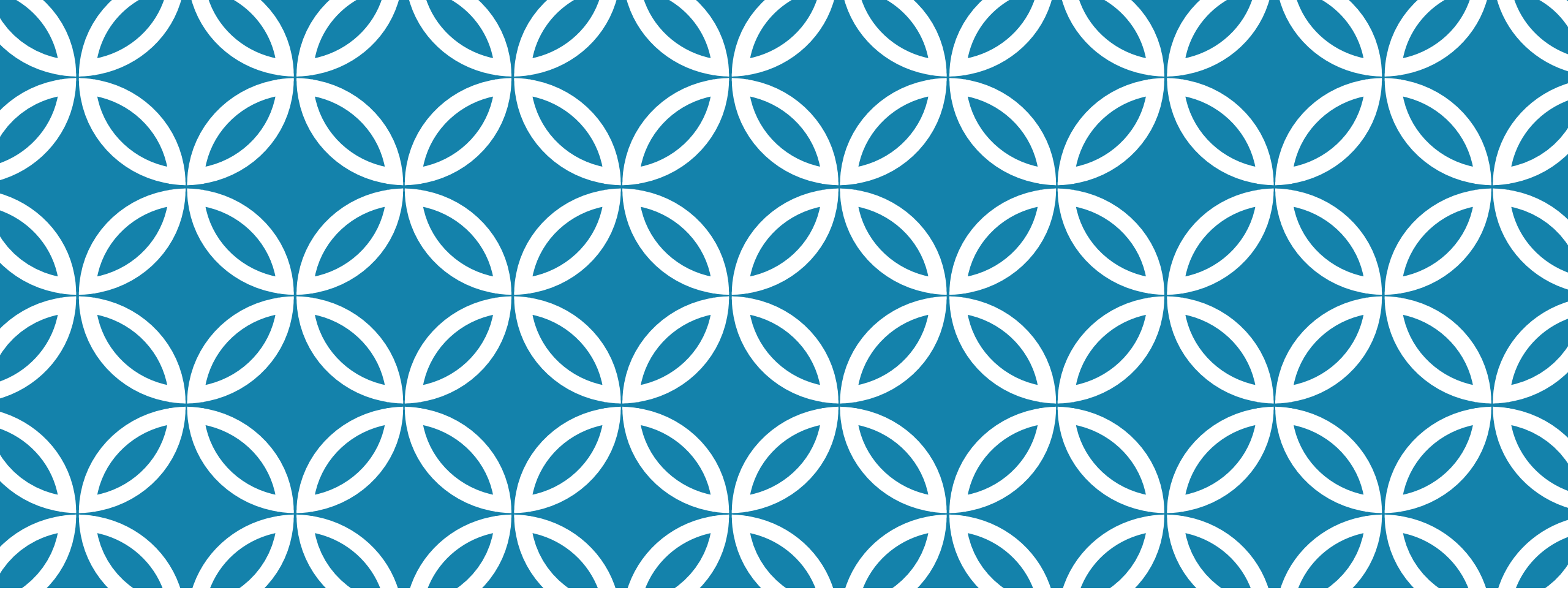


ADVANCED DATA STORAGE TECNIQUE IN PYTHON: LIST AND DICTIONARY

Safin Ahmmed
Lecturer,
CSE, KUET

Objectives

- Create, index, and slice a list
- Add and delete elements from a list
- Use list methods to append, sort, and reverse a list
- Use nested sequences to represent even more complex information
- Use dictionaries to work with pairs of data
- Add and delete dictionary items

Lists

- Lists
  - Sequences of any type
  - Like tuples, but mutable (can be modified)
  - Essentially can do everything tuples can, plus more

Creating a List

- **List:** A mutable sequence of any type
- Creating an Empty List

```
inventory = []
```

- Creating a List with Elements

```
inventory = ["sword", "armor", "shield",
             "healing potion"]
```

Using `len()` and `in` with Lists

- The len() function with lists
  - Just as with tuples, returns number of elements

```
print "You have", len(inventory), "items."
```

- The in operator with lists
  - Just as with tuples, tests for element membership

```
if "healing potion" in inventory:
   print "You will live to fight another day."
```

Indexing and Slicing Lists

- Indexing lists
  - Just as with tuples, supply the position number of the element in brackets

```
print "At index", index, "is",inventory[index]
```

- Slicing lists
  - Just as with tuples, supply the two end points, separated by a colon, in brackets

```
print inventory[begin:end]
```

Concatenating Lists

```
>>> inventory = ["sword", "armor", "shield",
                 "healing potion"]
>>> chest = ["gold", "gems"]
>>> inventory += chest
>>> print inventory
['sword', 'armor', 'shield', 'healing potion', 'gold',
  'gems']
```

- Just as with tuples, concatenation operator, +, works with lists

Understanding List Mutability

- **Mutable:** Changeable
- Lists are mutable
  - Elements (or slices) can be added
  - Elements (or slices) can be removed

Assigning a New List Element by Index

```
>>> inventory = ["sword", "armor", "shield",
                        "healing potion“, "gold", "gems"]
>>> inventory[0] = "crossbow”
>>> print inventory
 [' crossbow', 'armor', 'shield', 'healing potion', 'gold',
   'gems']
```

- Just as with tuples, you can assign a value to an existing list element

Assigning a New List Slice

```
>>> inventory = ["crossbow", "armor", "shield",
                 "healing potion“, "gold", "gems"]
>>> inventory[4:6] = ["orb of future telling"]
>>> print inventory
 ['crossbow', 'armor', 'shield', 'healing potion', 'orb of
   future telling’]
```

- Assignment statement replaces elements in slice with new element
- Replaces the two elements `inventory[4]` and `inventory[5]` with

  `"orb of future telling"`

Deleting a List Element

```
>>> inventory = ["crossbow", "armor", "shield",
                 "healing potion”, "orb of future telling"]
>>> del inventory[2]
>>> print inventory
 ['crossbow', 'armor', 'healing potion', 'orb of future
   telling’]
```

- Designate element to delete after `del`

Deleting a List Slice

```
>>> inventory = ["crossbow", "armor", "shield",
                 "healing potion”, "orb of future telling"]
>>> del inventory[:2]
>>> print inventory
 ['healing potion', 'orb of future telling’]
```

- Designate slice to delete after `del`

Inventory Program

```
inventory = ["sword",
             "armor",
             "shield",
             "healing potion"]

print ("Your items:\t")
for item in inventory:
    print (item)
input ("Press Enter key to continue")

# get the length of items in inventory

print ("You have ", len(inventory), "in your inventory")

input ("\nPress Enter key to continue")

# test for membership using in operator

if "healing potion" in inventory:
    print ("\nYou will live to fight another day!")


# display one item through an index

index = int(input("\nEnter an index number for an item in inventory:\t"))

print ("In index, ", index, "is:\t", inventory[(index -1)])

## display a slice
start = int(input("\nEnter an index number to begin slice:\t"))
end = int(input("\nEnter an index number to end slice:\t"))

print ("inventory[", start, ":", end, "] is:", end = "")
print (inventory[start:end])
input ("\nPress Enter to continue")

## concatenate two lists

chest = ["gold", "gems"]

print ("You find a chest which contains:\t")
print (chest)
print ("\nYou add the contents of the chest to your inventory")
inventory += chest
print ("\nYour inventory is now:")
print (inventory)
```

```
Console 1/A
Your items:
sword
armor
shield
healing potion

Press Enter key to continue
You have  4 in your inventory


Press Enter key to continue

You will live to fight another day!


Enter an index number for an item in inventory: 2
In index,  2 is:     armor


Enter an index number to begin slice:   2


Enter an index number to end slice: 4
inventory[ 2 : 4 ] is:['shield', 'healing potion']


Press Enter to continue
You find a chest which contains:
['gold', 'gems']

You add the contents of the chest to your inventory

Your inventory is now:
['sword', 'armor', 'shield', 'healing potion', 'gold', 'gems']
```

Inventory Program

```
inventory = ["sword",
             "armor",
             "shield",
             "healing potion"]

print ("Your items:\t")
for item in inventory:
    print (item)
input ("Press Enter key to continue")

# get the length of items in inventory

print ("You have ", len(inventory), "in your inventory")

input ("\nPress Enter key to continue")

# test for membership using in operator

if "healing potion" in inventory:
    print ("\nYou will live to fight another day!")


# display one item through an index

index = int(input("\nEnter an index number for an item in inventory:\t"))

print ("In index, ", index, "is:\t", inventory[(index -1)])

## display a slice
start = int(input("\nEnter an index number to begin slice:\t"))
end = int(input("\nEnter an index number to end slice:\t"))

print ("inventory[", start, ":", end, "] is:", end = "")
print (inventory[start:end])
input ("\nPress Enter to continue")

## concatenate two lists

chest = ["gold", "gems"]

print ("You find a chest which contains:\t")
print (chest)
print ("\nYou add the contents of the chest to your inventory")
inventory += chest
print ("\nYour inventory is now:")
print (inventory)
```

```
Console 1/A
Your items:
sword
armor
shield
healing potion

Press Enter key to continue
You have  4 in your inventory


Press Enter key to continue

You will live to fight another day!


Enter an index number for an item in inventory: 2
In index,  2 is:     armor


Enter an index number to begin slice:   2


Enter an index number to end slice: 4
inventory[ 2 : 4 ] is:['shield', 'healing potion']


Press Enter to continue
You find a chest which contains:
['gold', 'gems']

You add the contents of the chest to your inventory

Your inventory is now:
['sword', 'armor', 'shield', 'healing potion', 'gold', 'gems']
```

Inventory Program

```
inventory = ["sword",
             "armor",
             "shield",
             "healing potion"]

print ("Your items:\t")
for item in inventory:
    print (item)
input ("Press Enter key to continue")

# get the length of items in inventory

print ("You have ", len(inventory), "in your inventory")

input ("\nPress Enter key to continue")

# test for membership using in operator

if "healing potion" in inventory:
    print ("\nYou will live to fight another day!")


# display one item through an index

index = int(input("\nEnter an index number for an item in inventory:\t"))

print ("In index, ", index, "is:\t", inventory[(index -1)])

## display a slice
start = int(input("\nEnter an index number to begin slice:\t"))
end = int(input("\nEnter an index number to end slice:\t"))

print ("inventory[", start, ":", end, "] is:", end = "")
print (inventory[start:end])
input ("\nPress Enter to continue")

## concatenate two lists

chest = ["gold", "gems"]

print ("You find a chest which contains:\t")
print (chest)
print ("\nYou add the contents of the chest to your inventory")
inventory += chest
print ("\nYour inventory is now:")
print (inventory)
```

```
Console 1/A
Your items:
sword
armor
shield
healing potion

Press Enter key to continue
You have  4 in your inventory


Press Enter key to continue

You will live to fight another day!


Enter an index number for an item in inventory: 2
In index,  2 is:     armor


Enter an index number to begin slice:   2


Enter an index number to end slice: 4
inventory[ 2 : 4 ] is:['shield', 'healing potion']


Press Enter to continue
You find a chest which contains:
['gold', 'gems']

You add the contents of the chest to your inventory

Your inventory is now:
['sword', 'armor', 'shield', 'healing potion', 'gold', 'gems']
```

Inventory Program

```
inventory = ["sword",
             "armor",
             "shield",
             "healing potion"]

print ("Your items:\t")
for item in inventory:
    print (item)
input ("Press Enter key to continue")

# get the length of items in inventory

print ("You have ", len(inventory), "in your inventory")

input ("\nPress Enter key to continue")

# test for membership using in operator

if "healing potion" in inventory:
    print ("\nYou will live to fight another day!")


# display one item through an index

index = int(input("\nEnter an index number for an item in inventory:\t"))

print ("In index, ", index, "is:\t", inventory[(index -1)])

## display a slice
start = int(input("\nEnter an index number to begin slice:\t"))
end = int(input("\nEnter an index number to end slice:\t"))

print ("inventory[", start, ":", end, "] is:", end = "")
print (inventory[start:end])
input ("\nPress Enter to continue")

## concatenate two lists

chest = ["gold", "gems"]

print ("You find a chest which contains:\t")
print (chest)
print ("\nYou add the contents of the chest to your inventory")
inventory += chest
print ("\nYour inventory is now:")
print (inventory)
```

```
Console 1/A
Your items:
sword
armor
shield
healing potion

Press Enter key to continue
You have  4 in your inventory


Press Enter key to continue

You will live to fight another day!


Enter an index number for an item in inventory: 2
In index,  2 is:     armor


Enter an index number to begin slice:   2


Enter an index number to end slice: 4
inventory[ 2 : 4 ] is:['shield', 'healing potion']


Press Enter to continue
You find a chest which contains:
['gold', 'gems']

You add the contents of the chest to your inventory

Your inventory is now:
['sword', 'armor', 'shield', 'healing potion', 'gold', 'gems']
```

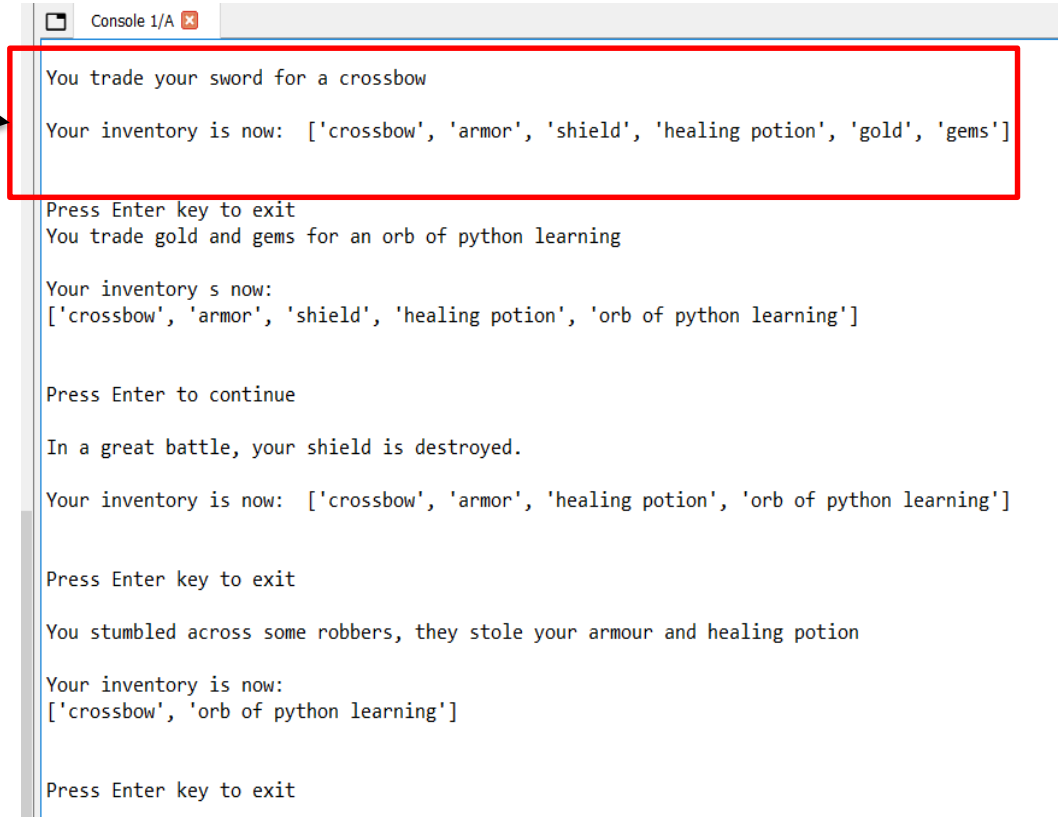
Inventory Program

```
inventory = ["sword",
             "armor",
             "shield",
             "healing potion"]

print ("Your items:\t")
for item in inventory:
    print (item)
input ("Press Enter key to continue")

# get the length of items in inventory

print ("You have ", len(inventory), "in your inventory")

input ("\nPress Enter key to continue")

# test for membership using in operator

if "healing potion" in inventory:
    print ("\nYou will live to fight another day!")


# display one item through an index

index = int(input("\nEnter an index number for an item in inventory:\t"))

print ("In index, ", index, "is:\t", inventory[(index -1)])

## display a slice
start = int(input("\nEnter an index number to begin slice:\t"))
end = int(input("\nEnter an index number to end slice:\t"))

print ("inventory[", start, ":", end, "] is:", end = "")
print (inventory[start:end])
input ("\nPress Enter to continue")

## concatenate two lists

chest = ["gold", "gems"]

print ("You find a chest which contains:\t")
print (chest)
print ("\nYou add the contents of the chest to your inventory")
inventory += chest
print ("\nYour inventory is now:")
print (inventory)
```

```
Console 1/A
Your items:
sword
armor
shield
healing potion

Press Enter key to continue
You have  4 in your inventory


Press Enter key to continue

You will live to fight another day!


Enter an index number for an item in inventory: 2
In index,  2 is:     armor


Enter an index number to begin slice:   2


Enter an index number to end slice: 4
inventory[ 2 : 4 ] is:['shield', 'healing potion']


Press Enter to continue
You find a chest which contains:
['gold', 'gems']

You add the contents of the chest to your inventory

Your inventory is now:
['sword', 'armor', 'shield', 'healing potion', 'gold', 'gems']
```

Inventory Program

```
# assign by index

print ("\nYou trade your sword for a crossbow")
inventory [0] = "crossbow"
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")

# assign by slice

print ("You trade gold and gems for an orb of python learning")
inventory [4:6] = ["orb of python learning"]
print ("\nYour inventory s now:\t")
print (inventory)

input ("\nPress Enter to continue")

# delete an element

print ("\nIn a great battle, your shield is destroyed.")
del inventory [2]
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")


print ("\nYou stumbled across some robbers, they stole your armour and \
healing potion")
del inventory [1:3]
print ("\nYour inventory is now:\t")
print (inventory)

input ("\nPress Enter key to exit")
```



Inventory Program

```
# assign by index

print ("\nYou trade your sword for a crossbow")
inventory [0] = "crossbow"
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")

# assign by slice

print ("You trade gold and gems for an orb of python learning")
inventory [4:6] = ["orb of python learning"]
print ("\nYour inventory s now:\t")
print (inventory)

input ("\nPress Enter to continue")

# delete an element

print ("\nIn a great battle, your shield is destroyed.")
del inventory [2]
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")


print ("\nYou stumbled across some robbers, they stole your armour and \
healing potion")
del inventory [1:3]
print ("\nYour inventory is now:\t")
print (inventory)

input ("\nPress Enter key to exit")
```

```
Console 1/A

You trade your sword for a crossbow

Your inventory is now:  ['crossbow', 'armor', 'shield', 'healing potion', 'gold', 'gems']


Press Enter key to exit
You trade gold and gems for an orb of python learning

Your inventory s now:
['crossbow', 'armor', 'shield', 'healing potion', 'orb of python learning']


Press Enter to continue

In a great battle, your shield is destroyed.

Your inventory is now:  ['crossbow', 'armor', 'healing potion', 'orb of python learning']


Press Enter key to exit

You stumbled across some robbers, they stole your armour and healing potion

Your inventory is now:
['crossbow', 'orb of python learning']


Press Enter key to exit
```

Inventory Program

```
# assign by index

print ("\nYou trade your sword for a crossbow")
inventory [0] = "crossbow"
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")

# assign by slice

print ("You trade gold and gems for an orb of python learning")
inventory [4:6] = ["orb of python learning"]
print ("\nYour inventory s now:\t")
print (inventory)

input ("\nPress Enter to continue")

# delete an element

print ("\nIn a great battle, your shield is destroyed.")
del inventory [2]
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")


print ("\nYou stumbled across some robbers, they stole your armour and \
healing potion")
del inventory [1:3]
print ("\nYour inventory is now:\t")
print (inventory)

input ("\nPress Enter key to exit")
```

```
Console 1/A

You trade your sword for a crossbow

Your inventory is now:  ['crossbow', 'armor', 'shield', 'healing potion', 'gold', 'gems']


Press Enter key to exit
You trade gold and gems for an orb of python learning

Your inventory s now:
['crossbow', 'armor', 'shield', 'healing potion', 'orb of python learning']


Press Enter to continue

In a great battle, your shield is destroyed.

Your inventory is now:  ['crossbow', 'armor', 'healing potion', 'orb of python learning']


Press Enter key to exit

You stumbled across some robbers, they stole your armour and healing potion

Your inventory is now:
['crossbow', 'orb of python learning']


Press Enter key to exit
```

Inventory Program

```
# assign by index

print ("\nYou trade your sword for a crossbow")
inventory [0] = "crossbow"
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")

# assign by slice

print ("You trade gold and gems for an orb of python learning")
inventory [4:6] = ["orb of python learning"]
print ("\nYour inventory s now:\t")
print (inventory)

input ("\nPress Enter to continue")

# delete an element

print ("\nIn a great battle, your shield is destroyed.")
del inventory [2]
print ("\nYour inventory is now:\t", end = "")
print (inventory)

input ("\nPress Enter key to exit")


print ("\nYou stumbled across some robbers, they stole your armour and \
healing potion")
del inventory [1:3]
print ("\nYour inventory is now:\t")
print (inventory)

input ("\nPress Enter key to exit")
```

```
Console 1/A

You trade your sword for a crossbow

Your inventory is now:  ['crossbow', 'armor', 'shield', 'healing potion', 'gold', 'gems']


Press Enter key to exit
You trade gold and gems for an orb of python learning

Your inventory s now:
['crossbow', 'armor', 'shield', 'healing potion', 'orb of python learning']


Press Enter to continue

In a great battle, your shield is destroyed.

Your inventory is now:  ['crossbow', 'armor', 'healing potion', 'orb of python learning']

Press Enter key to exit

You stumbled across some robbers, they stole your armour and healing potion

Your inventory is now:
['crossbow', 'orb of python learning']


Press Enter key to exit
```

Using List Methods

- **List** methods manipulate lists
- Through list methods, you can:
  - Add an element
  - Remove an element
  - Sort a list
  - Reverse a list

Selected List Methods

TABLE 5.1 SELECTED LIST METHODS

Method	Description
`append(value)`	Adds *value* to end of a list.
`sort()`	Sorts the elements, smallest value first.
`reverse()`	Reverses the order of a list.
`count(value)`	Returns the number of occurrences of *value*.
`index(value)`	Returns the first position number where *value* occurs.
`insert(i, value)`	Inserts *value* at position *i*.
`pop([i])`	Returns value at position *i* and removes value from the list. Providing the position number *i* is optional. Without it, the last element in the list is removed and returned.
`remove(value)`	Removes the first occurrence of *value* from the list.

The List `append()` Method

```
# Adds List Element as value of List.
List = ['Mathematics', 'chemistry', 1997, 2000]
List.append(20544)
print(List)
```

The List `append()` Method

```
# Adds List Element as value of List.
List = ['Mathematics', 'chemistry', 1997, 2000]
List.append(20544)
print(List)
```

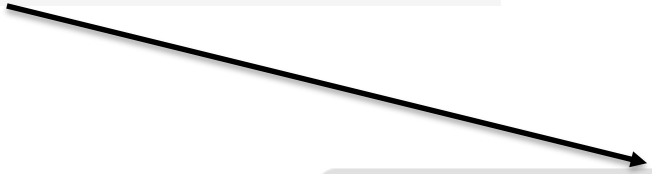
```
['Mathematics', 'chemistry', 1997, 2000, 20544]
```

The List `sort()` Method

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]

#Reverse flag is set True
List.sort(reverse=True)

#List.sort().reverse(), reverses the sorted list
print(List)
```

The List `sort()` Method

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]

#Reverse flag is set True
List.sort(reverse=True)

#List.sort().reverse(), reverses the sorted list
print(List)
```



```
[5.33, 4.445, 3, 2.5, 2.3, 1.054]
```

Using Nested Sequences

- **Nested Sequence:** A sequence inside another sequence
- A list can contain lists or tuples
- A tuple can contain tuples or lists

Creating Nested Sequences

```
>>> scores = [("Moe", 1000), ("Larry", 1500),
              ("Curly", 3000)]
>>> print scores
[('Moe', 1000), ('Larry', 1500), ('Curly', 3000)]
```

- scores is a nested sequence
- scores is a list of tuples
- scores has three elements, each of which is a tuple

Accessing Nested Elements

```
>>> scores = [("Moe", 1000), ("Larry", 1500),
              ("Curly", 3000)]
>>> print scores[2]
('Curly', 3000)
>>> print scores[2][0]
Curly
```

- scores[2] is the element of the list at position 2
- scores[2][0] is the element at position 0 of scores[2]

Unpacking a Sequence

```
>>> name, score = ("Shemp", 175)
>>> print name
Shemp
>>> print score
175
```

- **Sequence unpacking:** Automatically accessing each element of a sequence
- The tuple is unpacked as result of assignment statement

Accessing Elements of a Nested Sequence

```
for entry in scores:
    score, name = entry
    print name, "\t", score
```

- `entry` is an element of `scores`
- Assignment statement unpacks `entry`
- `score` is assigned first element of `entry`
- `name` is assigned second element of `entry`

Appending Elements to a Nested Sequence

```
entry = (score, name)
scores.append(entry)
```

- `append()` method works for any list, including a list of sequences
- New tuple `entry` is created
- `entry` is appended to list `scores` as last element

Shared References

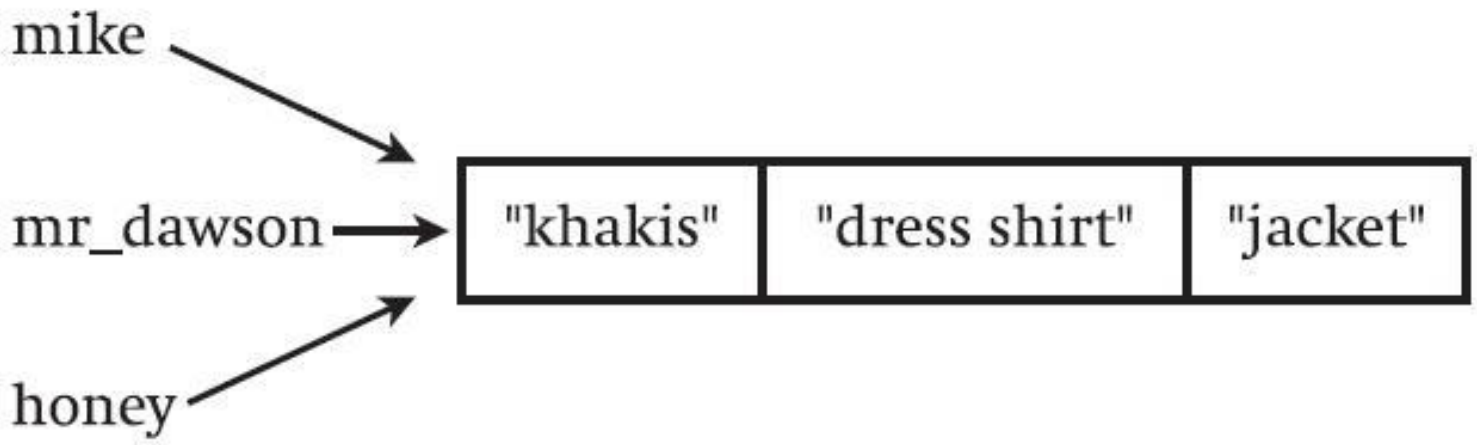


Figure 5.9: A single object has three references to it.

Mike, mr_dawson and honey all refer to same single list.

Shared References

```
>>> mike = ["khakis", "dress shirt", "jacket"]
>>> mr_dawson = mike
>>> honey = mike
>>> print mike
['khakis', 'dress shirt', 'jacket']
>>> print mr_dawson
['khakis', 'dress shirt', 'jacket']
>>> print honey
['khakis', 'dress shirt', 'jacket’]
```

All variables refer to same single list

Shared References

```
>>> honey[2] = "red sweater"
>>> print honey
['khakis', 'dress shirt', 'red sweater']
>>> print mike
['khakis', 'dress shirt', 'red sweater']
>>> print mr_dawson
['khakis', 'dress shirt', 'red sweater’]
```

- Change to list through one variable reflects change for all variables because there is only one list

Shared References

```
>>> mike = ["khakis", "dress shirt", "jacket"]
>>> honey = mike[:]
>>> honey[2] = "red sweater"
>>> print honey
['khakis', 'dress shirt', 'red sweater']
>>> print mike
['khakis', 'dress shirt', 'jacket']
```

- List slicing can create a new copy of a list and avoid shared references

Using Dictionaries

- **Dictionary:** A mutable collection of key-value pairs
- Like tuple and list, dictionary is another built-in type
- Unlike tuples and lists, dictionaries don’t organize data into sequences, but pairs
- Works like actual dictionary; look up one thing to get another
- Look up a key to get a value

Creating Dictionaries

```
geek = {"404" : "clueless.",
        "Uninstalled" : "being fired."}
```

- Creates new dictionary called `geek`
- `geek` has two entries or **items** (or elements)
- Each item is made up of a key and a value
- `404` is a key of one item; use it to look up value `"clueless."`
- Create dictionary by pairing values with colon, separated by commas, surrounded by curly braces

Using a Key to Retrieve a Value

```
>>> geek["404"]
'clueless.'
>>> geek["Uninstalled"]
'being fired.'
```

- Use key as index to get value
- Cannot use value as index to get key
- Using non-existent key as index produces error
- Dictionaries don't have position numbers – no order

Testing for a Key with the `in` Operator

```
>>> if "Dancing Baloney" in geek:
        print "I know what Dancing Baloney is."
else:
        print "I have no idea what Dancing Baloney is."

I have no idea what Dancing Baloney is.
```

- Use the `in` operator to test for key
- Condition is `True` if key exists in dictionary, `False` otherwise
- `in` operator can't be used to test for dictionary values

The Dictionary `get()` Method

```
>>> geek.get("404")
'clueless.'
>>> geek.get("Dancing Baloney")
None
>>> geek.get("Dancing Baloney", "I have no idea.")
'I have no idea.’
```

- Used for retrieving value based on key
- Has built-in safety net for handling non-existent key
  - If key exists, returns associated value
  - If key doesn’t exist, returns a default, program-provided value (or None if no default is provided)

Adding a Key-Value Pair

```
geek["Link Rot"] = "process by which web page links
  become obsolete."
```

- Dictionaries are mutable
- Add item by assigning value to dictionary indexed by key
- Overwrites current entry if key already exists in dictionary

Deleting a Key-Value Pair

```
del geek["404"]
```

- Removes key-value pair if key exists
- Generates error if key doesn’t exist

Geek Translator Program

```
geek = {"404": "clueless.  From the web error message 404, meaning page not found.",
        "Googling": "searching the Internet for background information on a person.",
        "Keyboard Plaque" : "the collection of debris found in computer keyboards.",
        "Link Rot" : "the process by which web page links become obsolete.",
        "Percussive Maintenance" : "the act of striking an electronic device to make it work ",
        "Uninstalled" : "being fired.  Especially popular during the dot-bomb era."}

choice = None
while choice != "0":

    print(
    """
    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term
    """
    )

    choice = input("Choice: ")
    print()

    # exit
    if choice == "0":
        print("Good-bye.")

    # get a definition
    elif choice == "1":
        term = input("What term do you want me to translate?: ")
        if term in geek:
            definition = geek[term]
            print("\n", term, "means", definition)
        else:
            print("\nSorry, I don't know", term)
```

```
Console 1/A
Geek Translator

0 - Quit
1 - Look Up a Geek Term
2 - Add a Geek Term
3 - Redefine a Geek Term
4 - Delete a Geek Term


Choice: 1


What term do you want me to translate?: 404

 404 means clueless.  From the web error message 404, meaning page not found

    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 2


What term do you want me to add?: Dancing Baloney


What's the definition?: I have no idea

 Dancing Baloney has been added.
```

Geek Translator Program

```
geek = {"404": "clueless.  From the web error message 404, meaning page not found.",
        "Googling": "searching the Internet for background information on a person.",
        "Keyboard Plaque" : "the collection of debris found in computer keyboards.",
        "Link Rot" : "the process by which web page links become obsolete.",
        "Percussive Maintenance" : "the act of striking an electronic device to make it work.",
        "Uninstalled" : "being fired.  Especially popular during the dot-bomb era."}

choice = None
while choice != "0":

    print(
    """
    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term
    """
    )

    choice = input("Choice: ")
    print()

    # exit
    if choice == "0":
        print("Good-bye.")

    # get a definition
    elif choice == "1":
        term = input("What term do you want me to translate?: ")
        if term in geek:
            definition = geek[term]
            print("\n", term, "means", definition)
        else:
            print("\nSorry, I don't know", term)
```

```
Console 1/A
    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 1


What term do you want me to translate?: 404

 404 means clueless.  From the web error message 404, meaning page not found

    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 2


What term do you want me to add?: Dancing Baloney


What's the definition?: I have no idea

 Dancing Baloney has been added.
```

Geek Translator Program

```
    # add a term-definition pair
    elif choice == "2":
        term = input("What term do you want me to add?: ")
        if term not in geek:
            definition = input("\nWhat's the definition?: ")
            geek[term] = definition
            print("\n", term, "has been added.")
        else:
            print("\nThat term already exists!  Try redefining it.")

    # redefine an existing term
    elif choice == "3":
        term = input("What term do you want me to redefine?: ")
        if term in geek:
            definition = input("What's the new definition?: ")
            geek[term] = definition
            print("\n", term, "has been redefined.")
        else:
            print("\nThat term doesn't exist!  Try adding it.")

    # delete a term-definition pair
    elif choice == "4":
        term = input("What term do you want me to delete?: ")
        if term in geek:
            del geek[term]
            print("\nOkay, I deleted", term)
        else:
            print("\nI can't do that!", term, "doesn't exist in the dictionary.")

    # some unknown choice
    else:
        print("\nSorry, but", choice, "isn't a valid choice.")

input("\n\nPress the enter key to exit.")
```

```
Console 1/A
    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 1


What term do you want me to translate?: 404

 404 means clueless.  From the web error message 404, meaning page not found

    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 2


What term do you want me to add?: Dancing Baloney


What's the definition?: I have no idea

 Dancing Baloney has been added.
```

Geek Translator Program

```
    # add a term-definition pair
    elif choice == "2":
        term = input("What term do you want me to add?: ")
        if term not in geek:
            definition = input("\nWhat's the definition?: ")
            geek[term] = definition
            print("\n", term, "has been added.")
        else:
            print("\nThat term already exists!  Try redefining it.")

    # redefine an existing term
    elif choice == "3":
        term = input("What term do you want me to redefine?: ")
        if term in geek:
            definition = input("What's the new definition?: ")
            geek[term] = definition
            print("\n", term, "has been redefined.")
        else:
            print("\nThat term doesn't exist!  Try adding it.")

    # delete a term-definition pair
    elif choice == "4":
        term = input("What term do you want me to delete?: ")
        if term in geek:
            del geek[term]
            print("\nOkay, I deleted", term)
        else:
            print("\nI can't do that!", term, "doesn't exist in the dictionary.")

    # some unknown choice
    else:
        print("\nSorry, but", choice, "isn't a valid choice.")

input("\n\nPress the enter key to exit.")
```

```
    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 4


What term do you want me to delete?: Dancing Baloney

Okay, I deleted Dancing Baloney

    Geek Translator

    0 - Quit
    1 - Look Up a Geek Term
    2 - Add a Geek Term
    3 - Redefine a Geek Term
    4 - Delete a Geek Term


Choice: 0

Good-bye.
```

Dictionary Requirements

- Keys
  - Must be unique
  - Must be immutable
- Values
  - Can be mutable or immutable
  - Doesn’t have to be unique

Built-in Dictionary Functions:

SN	Function with Description
1	cmp(dict1, dict2) Compares elements of both dict.
2	len(dict) Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
3	str(dict) Produces a printable string representation of a dictionary
4	type(variable) Returns the type of the passed variable. If passed variable is dictionary then it would return a dictionary type.

Built-in Dictionary Methods:

SN	Methods with Description
1	dict.clear() Removes all elements of dictionary *dict*
2	dict.copy() Returns a shallow copy of dictionary *dict*
2	dict.fromkeys() Create a new dictionary with keys from seq and values *set* to *value*.
3	dict.get(key, default=None) For *key* key, returns value or default if key not in dictionary
4	dict.has_key(key) Returns *true* if key in dictionary *dict*, *false* otherwise
5	dict.items() Returns a list of *dict*'s (key, value) tuple pairs
6	dict.keys() Returns list of dictionary dict's keys
7	dict.setdefault(key, default=None) Similar to get(), but will set dict[key]=default if *key* is not already in dict
8	dict.update(dict2) Adds dictionary *dict2*'s key-values pairs to *dict*
9	dict.values() Returns list of dictionary *dict2*'s values

REFERENCES

- https://www.geeksforgeeks.org/python-list/
- https://www.tnstate.edu/faculty/fyao/COMP3050/Py-Slides-5.ppt