CS521 O2
Information Structures with Python

Lecture 4

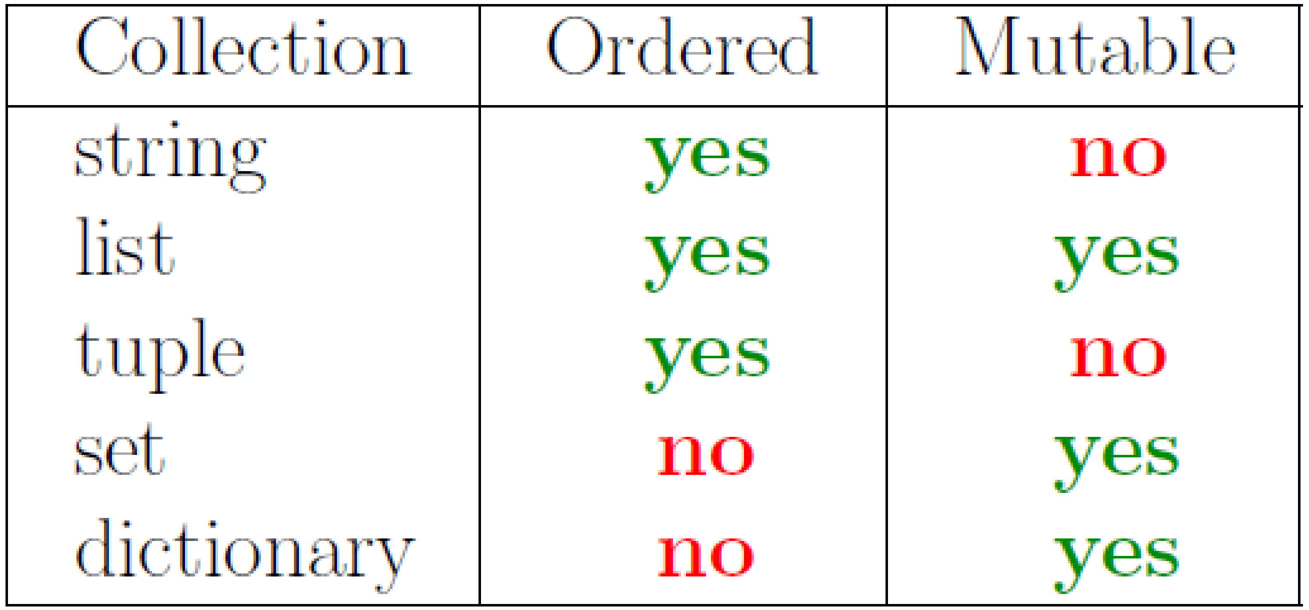
Guanglan Zhang

guanglan@bu.edu

Some slides adapted from Prof. Eugene Pinsky

Table of Content

More on mutability



Collection	Ordered	Mutable
string	yes	no
list	yes	yes
tuple	yes	no
set	no	yes
dictionary	no	yes

- Strings are immutable and can only contain immutable elements
- Tuples are immutable, but can contain mutable objects, which can be modified in place
- Sets are mutable. But the elements contained in a set must be of immutable type - lists and dictionaries cannot be included in sets
- Dictionary keys cannot contain mutable objects

Hashing

- Hash tables are used to implement map and set data structures in many common programming languages
- Python uses hash tables for dictionaries and sets
- Hash table is an unordered collection of key-value pairs, where each key is unique
- Hashing is the process of using an algorithm to map data of any size to a fixed length. This is called a hash value.
- Hashing is used to create high performance, direct access data structures where large amount of data can be stored and accessed quickly.
- Hash values are computed with hash functions

Hashable

- An object is hashable if it has a hash value which never changes during its lifetime
- It can have different values during multiple invocations of Python programs
- A hashable object needs a __hash__() method and an __eq__() method to perform comparison
- Hashable objects can be used as a dictionary keys and a set members because these data structures use the hash value internally
- Primitive types are hashable
- Immutable objects with immutable elements are hashable – strings, frozen set, and some tuples
- Mutable collections (such as lists or dictionaries) are not hashable
- Objects which are instances of user-defined classes are hashable by default; they all compare unequal, and their hash value is their id()

Function hash()

- Function hash() returns the hash value of the object if it has one
- (x == y) compares hash values
- (x is y) compares id values
- Hash values are integers
- They are used to quickly compare dictionary keys during a dictionary lookup
- Functions are hashable
- Hash values of custom classes are set to their ids. By default, all instances of custom classes will have a hash value defined at creation and it will not change over time. Two instances of the same class will have two different hash values.

Non-hashable Tuples

- A tuple is hashable only if all its elements are hashable.

- Tuples with mutable collections are not hashable
  - ✓ Iteration can be applied
  - ✓ In-place modification can be done – we may end up with tuples with “identical” collectoins

- A hashable tuple cannot contain lists, sets, or dictionaries

Copying in Python

- Assignment statements in Python do not create copies of objects, they only bind names to an object

- For mutable objects or collections of mutable objects, we may want to create real copies of these objects - we want copies that we can modify without the original being modified at the same time

- For sets, dictionaries, and lists, there is difference between shallow and deep copying:
  - ✓ A shallow copy constructs a new collection object and then populating it with references to the child objects found in the original.
  - ✓ A deep copy means first constructing a new collection object and then recursively populating it with copies of the child objects found in the original.

None object

- In many other languages, null is just a synonym for 0, but null in Python is a full-blown object
- None is a singleton - the NoneType class only ever gives you the same single instance of None
- Like True and False, None is an immutable keyword
- None is falsy, which means not None is True
- None is the value a function returns when there is no return statement in the function

Algorithms

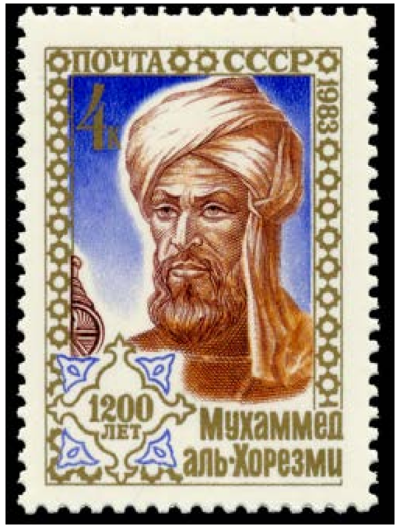
An **algorithm** is a procedure or a set of instructions used to perform or complete a particular task

- Finite set of instructions

- Origins from mathematics (Muhammad ibn Mūsā al-Khwārizmī, “father of algebra”)

- Lost in translation, the name of al-Khwārizmī became Algoritmi and changed meaning into “calculation methods”

- In mathematics and computer science, an algorithm is a step-by-step procedure for solving a problem or class of problems

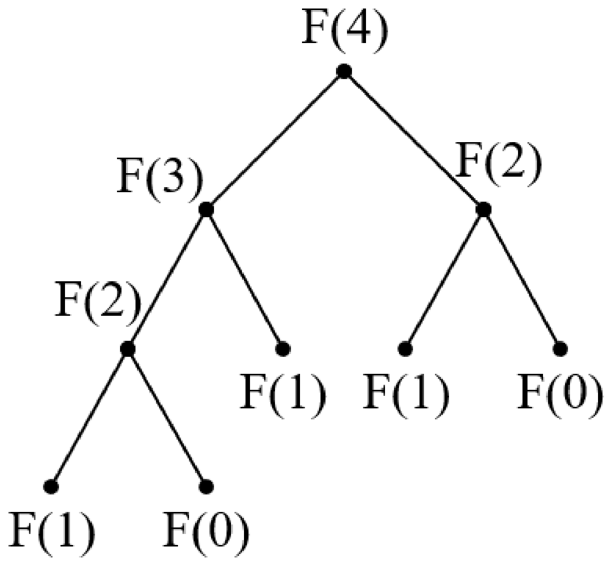


Using For loop or While loop to solve the problem

Fibonacci number: $Fn = F_{n-1} + F_{n-2}$ and $F_1 = 1$, $F_0 = 0$. **0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 56, ...**

```
START
Initialize an array $F[]$ with $F[0]=0$ and $F[1]=1$
Read user input an integer no less than 0 and save it in $n$
If $0 <= n <= 1$
        result = $F[n]$
Else if $n >= 2$
        Set $i=2$
        REPEAT
                $F[i]=F[i-1]+F[i-2]$
                $i=i+1$
        If $i > n$ end REPEAT
        result = $F[n]$
Else
        result = “The input $n$ must be an integer no less than 0”.
Endif

Output the result
END
```



Exercise:

- Give an example of two immutable objects x; y with the same hash value h and x == y is True

- Give an example of two immutable objects x; y with the same hash value h and x == y is False

- Which objects are hashable?

a. “seven”
b. {1:”A”, 2:”B”}
c. (1, 2, 3)
d. {1, 2, 3}
e. ([4, 5], 6)

Exercise:

- What will the result be?

```
x = [1 ,2 ,3]; y = x
print (id(x )== id(y))

x = [1 ,2 ,3]; y = [1, 2, 3]
print (id(x )== id(y))

x = [1 ,2 ,3]; y = x.copy()
print (id(x )== id(y))

x = [[1 ,2] ,3]; y = x. copy ()
x [0][0] = 100; print (x, y)
```

Key takeaways

- Strings are immutable and can only contain immutable elements
- Tuples are immutable, but can contain mutable objects, which can be modified in place
- Sets are mutable. But the elements contained in a set must be of immutable type
- Dictionary keys must be immutable objects
- An object is hashable if it has a hash value which never changes during its lifetime
- Primitive types are hashable
- Immutable objects with immutable elements are hashable – strings, frozen set, and some tuples
- Mutable collections (such as lists, sets, dictionaries) are not hashable