TCS332 Fundamental of Information Security and Blockchain

B. Tech CSE III Semester

Instructor:

Dr Mohammad Wazid

Professor, Department of CSE

Head of Cyber security and IoT research group

Graphic Era (Deemed to be University), Dehradun, India

Email: wazidkec2005@gmail.com

Homepage: https://sites.google.com/site/mwazidiiith/home

Python list, tuple, dictionary, functions and slicing

Python Functions

- A function is a block of code which only runs when it is called.
- We can pass data, known as parameters, into a function. A function can return data as a result.

Creating a Function

- In Python a function is defined using the def keyword:

Example

```
def my_function():
```

Calling a Function

To call a function, use the function name followed by parenthesis:

Example

```
def my_function():
  print(“Welcome to GEU")
my_function()
```

Output:

Welcome to GEU

Parameter passing

- The following example shows how to use a default parameter value.
- If we call the function without parameter, it uses the default value:

Example

```
def my_function(country = "Norway"):
  print("I am from " + country)
my_function("Sweden")
my_function("India")
my_function()
```

Output:

I am from Sweden

I am from India

I am from Norway

Passing a List as a Parameter

- You can send any data types of parameter to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.
- For example, if you send a List as a parameter, it will still be a List when it reaches the function:

Example

```
def my_function(food):
  for x in food:
    print(x)
fruits = ["apple", "banana", "cherry"]
my_function(fruits)
```

Output:
apple
banana
cherry

Return Values: To let a function return a value, use the return statement.

Example

```
def my_function(x):
  return 5 * x
print(my_function(7))
print(my_function(9))
print(my_function(10))
```

Output:

35

45

50

Recursion

- Python also accepts function recursion, which means a defined function can call itself.
- Recursion is a common mathematical and programming concept.
- It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

```
# An example of a recursive function to find the factorial of a number

def calc_factorial(x):
    if x == 1:
        return 1
    else:
        return (x * calc_factorial(x-1))

num = int(input("Enter the value of number:"))
print("The factorial of", num, "is", calc_factorial(num))
```

Output:

Enter the value of number:4

The factorial of 4 is 24

Function with loop

```
#With loop
def my_function():
    for i in range(5):
        print("Welcome to GEU")
my_function()
```

Output: Five times **Welcome to GEU**

Function with loop

```
def my_function():
    for i in range(5):
        print("the value of i is:", i)
        print("Welcome to GEU")
my_function()
```

Output:

```
the value of i is: 0
Welcome to GEU
the value of i is: 1
Welcome to GEU
the value of i is: 2
Welcome to GEU
the value of i is: 3
Welcome to GEU
the value of i is: 4
Welcome to GEU
```

#To print the Table of a given number. Use Time function to compute the execution time of the program.

```
import time
start = time.time()
def my_function():
    num = int(input("Enter the number"))
    for i in range(1, 11):
        tb = num*i
        print("The table of the entered number is:", tb)

my_function()
end = time.time()
tm=end - start
print("The execution time (sec) of the program is:",
tm)
```

Output:

```
Enter the number9
The table of the entered number is: 9
The table of the entered number is: 18
The table of the entered number is: 27
The table of the entered number is: 36
The table of the entered number is: 45
The table of the entered number is: 54
The table of the entered number is: 63
The table of the entered number is: 72
The table of the entered number is: 81
The table of the entered number is: 90
The execution time (sec) of the
program is: 3.88431715965271
```

Python Collection (Arrays)

- There are four collection data types:
- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

- **How to create a List:**

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Output:

```
['apple', 'banana', 'cherry']
```

- **Access Items:**
- You can access the list items by referring to the index number:
- Example: This will print the second item of the list:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Output:

```
banana
```

Change Item Value:

To change the value of a specific item, refer to the index number:

Example: Change the second item in the list:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

Output: This will replace banana with blackcurrant

['apple', 'blackcurrant', 'cherry']

Item Exists or not: To determine if a specified item is present in a list.

Example: Check if "apple" is present in the list:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

Output:

Yes, 'apple' is in the fruits list

List Length:

To determine how many items are there in the list. We can use len() method:

Example: Print the number of items in the list:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Output:

3

Add Items:

To add an item to the end of the list, use the append() method:

Example

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Output:

```
['apple', 'banana', 'cherry', 'orange']
```

To add an item at the specified index, use the insert() method:

Example: Insert an item as the second position:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

Output:

['apple', 'orange', 'banana', 'cherry']

Remove Item

Using remove() method.

Example: The remove() method removes the specified item:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Output:

['apple', 'cherry']

The pop() method removes the specified index, (or the last item if index is not specified):

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

Output:

['apple', 'banana']

If we specify index number:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(0)
print(thislist)
```

Output:

```
['banana', 'cherry']
```

The del keyword removes the specified index:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Output:

```
['banana', 'cherry']
```

The del keyword can also delete the list completely:

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Output:

Nothing will be printed. All list items are deleted.

Copy a List

There are ways to make a copy, one way is to use the built-in List method copy().

Example: Make a copy of a list with the copy() method:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist[:]
#mylist = thislist.copy() # only in some version of python
print(mylist)
```

Output:

['apple', 'banana', 'cherry']

The list() Constructor

It is also possible to use the list() constructor to make a new list.

Example: Using the list() constructor to make a List:

```
thislist = list(("apple", "banana", "cherry")) # Use double round-brackets
print(thislist)
```

Output:

```
['apple', 'banana', 'cherry']
```

Different List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements
extend()	Add the elements of a list to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

List sort() example:

```
# vowels list
vowels = ['e', 'a', 'u', 'o', 'i']
# sort the vowels
vowels.sort()
# print vowels
print('Sorted list:', vowels)
# print vowels in reverse order
vowels.reverse()
print('Reverse order list:', vowels)
```

Output:

```
Sorted list: ['a', 'e', 'i', 'o', 'u']
Reverse order list: ['u', 'o', 'i', 'e', 'a']
```

Tuple

A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets.

Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Output: ('apple', 'banana', 'cherry')

Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets:

Example: Return the item in position 1:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

Output: banana

Python Dictionary(Dict):

- Dictionaries are another example of a data structure.
- A dictionary is used to map or associate things you want to store the keys you need to get them.
- A dictionary in Python is just like a dictionary in the real world.
- Python Dictionary are defined into two elements Keys and Values.

Syntax for Python Dictionary:

Dict = { ' Tim': 18, xyz,.. }

Dictionary is listed in curly brackets, inside these curly brackets, keys and values are declared. Each key is separated from its value by a colon (:) while each element is separated by commas.

Properties of Dictionary Keys

- Important points.
- More than one entry per key is not allowed (no duplicate key is allowed)
- The values in the dictionary can be of any type while the keys must be immutable like numbers, strings.
- Dictionary keys are case sensitive-Same key name but with the different case are treated as different keys in Python dictionaries.

Example:

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
print (Dict['Tiffany'])
```

Output:

22

Here: Tiffany is key and 22 is key value.

In code, we have dictionary name "Dict"

We declared the name and age of the person in the dictionary, where name is "Keys" and age is the"value"

Python Dictionary Methods: Copying dictionary

You can also copy the entire dictionary to new dictionary. For example, here we have copied our original dictionary to new dictionary name "Boys" and "Girls".

Example:

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
Boys = {'Tim': 18,'Charlie':12,'Robert':25}
Girls = {'Tiffany':22}
studentX=Boys.copy()
studentY=Girls.copy()
print("Boys students are:", studentX)
print("Girls students are:", studentY)
```

Output:

Boys students are: {'Tim': 18, 'Charlie': 12, 'Robert': 25}

Girls students are: {'Tiffany': 22}

- We have the original dictionary (Dict) with the name and age of the boys and girls together.
- But we want boys list separate from girls list, so we defined the element of boys and girls in a separate dictionary name "Boys" and "Girls."
- We have created new dictionary name "studentX" and "studentY", where all the keys and values of boy dictionary are copied into studentX, and the girls will be copied in studentY.

Updating Dictionary: We can also update a dictionary by adding a new entry or by deleting an existing entry. Here in the example we will add another name "Sumit" to our existing dictionary.

Example

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

Dict.update({"Sumit":9})

print(Dict)
```

• Our existing dictionary "Dict" does not have the name "Sumit."

We use the method Dict.update to add Sumit to our existing dictionary.

Output:

{'Tim': 18, 'Charlie': 12, 'Tiffany': 22, 'Robert': 25, 'Sumit': 9}

Delete Keys from the dictionary

Python dictionary gives you the liberty to delete any element from the dictionary list. Suppose you don't want the name Charlie in the list, so you can delete the key element by following code.

Example

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}

del Dict ['Charlie']

print(Dict)
```

Output: {'Tim': 18, 'Tiffany': 22, 'Robert': 25}

This will print the dictionary list without Charlie.

Dictionary items() Method: The items() method returns a list of tuple pairs (Keys, Value) in the dictionary.

Example

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
print("Students Name: %s" % list(Dict.items()))
```

- We use the code items() method for our Dict.
- When code was executed, it returns a list of items (keys and values).

Output:

Students Name: [('Tim', 18), ('Charlie', 12), ('Tiffany', 22), ('Robert', 25)]

Sorting the Dictionary

- In the dictionary, we can also sort the elements.
- For example, if we want to print the name of the elements of our dictionary alphabetically.
- It will sort each element of dictionary accordingly.

Sorting the Dictionary

Example

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
Students = list(Dict.keys())
Students.sort()
for S in Students:
    print(":".join((S,str(Dict[S]))))
```

Sorting the Dictionary

Example

Output:

Charlie:12

Robert:25

Tiffany:22

Tim:18

Sorting the Dictionary example.

Explanation:

- We declared the variable students for our dictionary “Dict.”
- Then we use the code Students.sort, which will sort the element inside our dictionary.
- But to sort each element in dictionary, we run the for loop by declaring variable S.
- Now, when we execute the code the for loop will call each element from the dictionary, and it will print the string and value in an order

Dictionary len() Method (to compute the length)

The len() function gives the number of pairs in the dictionary.

```
Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
print("Length : %d" % len (Dict))
```

Output:

Length : 4

Dictionary methods summary:

Method	Description	Syntax
copy()	Copy the entire dictionary to new dictionary	dict.copy()
update()	Update a dictionary by adding a new entry or a key-value pair to an existing entry or by deleting an existing entry.	Dict.update()
items()	Returns a list of tuple pairs (Keys, Value) in the dictionary.	dictionary.items()
sort()	You can sort the elements	dictionary.sort()
len()	Gives the number of pairs in the dictionary.	len(dict)
cmp()	Compare values and keys of two dictionaries	cmp(dict1, dict2)

Python slice() Function

- The slice() function returns a slice object.
- A slice object is used to specify how to slice a sequence. You can specify where to start the slicing, and where to end. You can also specify the step, which allows you to e.g. slice only every other item.

Syntax

slice(start, end, step)

Example: Create a tuple and a slice object. Use the slice object to get only the two first items of the tuple:

```
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(2)
print(a[x])
```

Output:

('a', 'b')

Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start the slicing. Default is 0
end	An integer number specifying at which position to end the slicing
step	Optional. An integer number specifying the step of the slicing. Default is 1

Example: Create a tuple and a slice object. Start the slice object at position 3, and slice up to less than position 5, and return the result:

```
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(3, 5)
print(a[x])
```

Output:

('d', 'e')

Example 2:

```
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(3, 6)
print(a[x])
```

Output:

('d', 'e', 'f')

Example: Create a tuple and a slice object. Use the step parameter to return every third item:

```
a = ("a", "b", "c", "d", "e", "f", "g", "h")
x = slice(0, 8, 3)
print(a[x])
```

Output:

('a', 'd', 'g')

Negative Index to a Python List

- Index value at the end is -1. Which decreases when we move left in the list.

```
lst1 = ['Ajay', 'Bobby', 'Ashok', 'Vijay', 'Anil', 'Rahul', 'Alex', 'Christopher']

print(lst1[-2])
```

Output:

Alex

Slicing Python Lists

Slicing lists helps in fetching sections (slices) of the list. This is a very useful command to get a partial list from a parent list.

lst [start:end] # Items from index=start, to index=listlength-1 **(summary: all items)**

lst [start:] # Items index=start through the rest of the array **(summary: all items)**

lst [:end] # All items from the beginning through index=listlentgh-1 **(summary: all items)**

lst [:] # A copy of the whole array **(summary: all items)**

Applications: Slicing is very helpful in the analysis of generated log files which are used to detect any sign of intrusion in the network/system.

Example:

```
lst1 = ['Ajay', 'Bobby','Ashok', 'Vijay', 'Anil', 'Rahul','Alex', 'Christopher']
print(lst1[-2])
print(lst1[0:])
print(lst1[:8])
print(lst1[0:8])
print(lst1[-1])
print(lst1[0])
print(lst1[:])
```

Output:

- Alex
- ['Ajay', 'Bobby', 'Ashok', 'Vijay', 'Anil', 'Rahul', 'Alex', 'Christopher']
- ['Ajay', 'Bobby', 'Ashok', 'Vijay', 'Anil', 'Rahul', 'Alex', 'Christopher']
- ['Ajay', 'Bobby', 'Ashok', 'Vijay', 'Anil', 'Rahul', 'Alex', 'Christopher']
- Christopher
- Ajay
- ['Ajay', 'Bobby', 'Ashok', 'Vijay', 'Anil', 'Rahul', 'Alex', 'Christopher']

References

- Python programming information available at:

http://knowledgehills.com/python

- Python programming information available at:

https://www.programiz.com/python-programming

- Balagurusamy, “Problem Solving and Python Programming Paperback, TMH, 13 Sep 2017.
- Harsh Bhasin, "Python for Beginners Paperback", New Age International Publisher, 1 Jan 2018.
- Martin C. Brown, "Python: The Complete Reference", TMH, 20 Mar 2018.
- Mark Lutz, "Programming Python", 4th Edition, O'Reilly Media, December 2010
- Wesley J Chun , "Core Python Programming", Prentice Hall, 18 Sep 2006.