Department of Computer Technology and Information Systems

CTIS264 – Computer Algorithms

Spring 2019 - 2020

Lab Guide 2 - Week 3

Instructor : Füsun YÜRÜTEN

Assistant : Leyla SEZER

Objectives:

- **Python Collections**
- **Lists (sort, reverse, remove operations)**
- **Tuple**
- **Dictionary**
- **for loops**
- **if- else statement**
- **Strings, slicing strings**
- **Random number generation**
- **Timing**
- **Middle-School Procedure *for GCD and Sieve of Eratosthenes***

Python Collections : There are some collection data types in Python programming language:

- **List** is a collection which is ordered and **changeable**. Allows duplicate members. A list is denoted by square brackets [ ].
- **Tuple** is a collection which is ordered and **unchangeable**. Allows duplicate members. Tuples are represented using parentheses ( ).
- **Dictionary** is a collection which is unordered, **changeable** and indexed. No duplicate members are allowed. Dictionary objects are created using curly braces { }.

  Syntax:

  dict = {key1 : value1, key2 : value2, … keyN : valueN }

Q1. Python Lists (Arrays): Table shows the list methods to make some simple list operations.

List Methods

Python has a set of built-in methods that you can use on lists.

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 with the specified value
extend()	Add the elements of a list (or any iterable), 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

Use LIST methods to solve following questions.
Write a Python program that;

- Inputs 5 integer numbers from the user and put them into a list,
- Finds the average of these numbers, display the average,
- Sorts the list content and display,
- Reverse the list,
- Removes the first number from the list which is lower than the average,
- Displays the final format of the list.

Output:

```
Enter a number:9
Enter a number:5
Enter a number:4
Enter a number:7
Enter a number:6
Average is 6.2
List of numbers  [9, 5, 4, 7, 6]
Sorted format of the list  [4, 5, 6, 7, 9]
Reverse format of the list  [9, 7, 6, 5, 4]
Final format of list: [9, 7, 5, 4]
```

Q2. Python Tuples;

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

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

Output:

```
banana
```

- You can use negative indexing that means beginning from the end, -1 refers to the last item, -2 refers to the second last item, etc.

```
print(thistuple[-1])
```

Output:

"cherry".

- You can specify a range of indexes by specifying where to start and where to end the range.

```
thistuple =
("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

Output:

```
('cherry','orange','kiwi')
```

- Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called. But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)
print(x)
```

Output:

```
('apple','kiwi','cherry')
```

- The del keyword can delete the tuple completely. You cannot remove items in a tuple.
- You can concatenate two tuple by using '+' sign. Ex; `tuple3= tuple1 + tuple2`.

Tuple Methods

Python has two built-in methods that you can use on tuples.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Write Python program that;

- Creates a list of tuples containing the course name and credits of some courses.
  - Courses: ('CS 125',3),('HIST 200',4),('PHIL 243',6),('POLS 304',3), ('ENG 101',3)
- Calculates and displays the total credits for all first year courses in the list. You may assume that all course codes are exactly 3 digits.

Output:

```
First year courses:
CS 125
ENG 101

Total credits for first year courses: 6
```

Q3. Python Dictionaries;

- Example of dictionary creation.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Output:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

- You can access the items of a dictionary by referring to its key name, inside square brackets or using get() method. Ex; `thisdict["model"]` or `thisdict.get("model")`
- You can change the value of a specific item by referring to its key name.

```
thisdict["year"] = 2018
```

- You can print all *values* in the dictionary, one by one.

```
for x in thisdict:
  print(thisdict[x])
```

- Loop through both *keys* and *values*, by using the items() mehod.

```
for x, y in thisdict.items():
  print(x, y)
```

- To determine if a specified key is present in a dictionary use the in keyword.

```
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

- You can create a dictionary that contain three dictionaries, nested dictionaries.

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

Output:

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias',
'year': 2007}, 'child3': {'name': 'Linus', 'year': 2011}}
```

Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Write a Python program that;

- Creates a dictionary where the keys are the brand and the model of cars, and the values are the price of the cars. The dictionary should contain the cars and prices shown in the output below.
- Display the dictionary.
- Display the cars whose prices are over 100000.
- Change prices of cars which are lower than 100000, to 99000.
- Display the final dictionary.

Output:

```
Original Dictionary:
{'Honda CR-V': 125000, 'Volkswagen Passat': 135000, 'Toyota Yaris': 55000,
'Volkswagen Toureg': 255000, 'Honda Civic': 95000}

Prices over 100000:
Honda CR-V 125000
Volkswagen Passat 135000
Volkswagen Toureg 255000

Updated Dictionary:
{'Honda CR-V': 125000, 'Volkswagen Passat': 135000, 'Toyota Yaris': 99000,
'Volkswagen Toureg': 255000, 'Honda Civic': 99000}
```

Q4.

Write a Python program that;

- Gets the name of the user, then generates a password for that user, the rule is;
  - Get the characters of the name (form 3 to 6) then put a random number between (1,900) end of the subtracted string.

Hint: For the random number use the following code segment;

```
import random
rand_num = random.randint(1,900)
```

Output:

```
Enter your name and surname to password: GizemOnses
GizemOnses
Slicing the name (3,6): emOn
Random number (1-900): 771
Generated password emOn771
```

Q5.

Analysis of the algorithms with empirical analysis;

- Select a specific (typical) sample of inputs.
- Use physical unit of time (e.g., milliseconds).
- Analyze the empirical data.

Write a Python program that;

- Finds the sum of numbers between 0 and 100000, within for loop. Write a function that calculates this sum and displays the execution time for this for loop (empirical analysis for time efficiency of the function). And call this function at least 5 times to see the time differences.
- In the main body of the program, find the result of the following formula (which is the formula to find the sum of the n consecutive numbers), by considering the time efficiency with empirical analysis. Give 100000 to n.

$$\sum_{i=0}^{n} \frac{n(n+1)}{2}$$

Output:

```
Sum is 5000050000 required  0.0065024 seconds
Sum is 5000050000 required  0.0156236 seconds
Sum is 5000050000 required  0.0156248 seconds
Sum is 5000050000 required  0.0156245 seconds
Sum is 5000050000 required  0.0156243 seconds
Sum is 5000050000.0  required 0.0 seconds if we use formulae
```

Q6. Write a Python program that;

- Writes a function, ***Sieve of Eratosthenes*** as defined in chapter 1 (CH01 in lecture notes) by using python. Do not forget that mathematicians do not consider 1 to be a prime number. So output of this function should not include 1. Also, in order to able to use the output array of this function in the ***prime factorization*** function defined in the 2nd part of the question, you have to eliminate all zero (0) values from output array.
- Writes a function, ***prime factorization*** which will find and return all the prime factors of a given positive integer number. This function should call and use the output of ***Sieve of Eratosthenes*** function while finding prime factorization of the given number.

  Middle-school procedure;

  - Step 1: Find the prime factorization of *m.*
  - Step 2: Find the prime factorization of *n.*
  - Step 3: Find all the common prime factors of this two numbers and print them.
  - Step 4: Compute the product of all the common prime factors and print it as gcd*(m,n*)
- Then program finds and prints two different positive integers (let's say m & n) ***greatest common divisor*** by using middle school procedure as mentioned above.

Hint: *You may* ***import math*** *library and you may use* ***math.sqrt(***x***)*** *and* ***math.floor(***x***)*** *functions in python.*

Output:

```
Non-negative integer >=2 please: 685000
Another Non-negative integer >=2 please: 4712
Prime factors of n  [2, 2, 2]
Prime factors of m  [2, 2, 2]

Common prime factors are: [2, 2, 2]
Greatest common divisor: 8
Passed time in execution: 0.43109560012817383
```