Data Collections and Random Numbers

CS 177 – Recitation 9

Exam 1

- Exam key is posted in wiki. Check the solution and learn.
- Average: 63.54
- Median: 64

Question: is the class final grade curved?

Answer: No, your final grade consists of:

Lab 25% + Project 25% + 3 Exams 50%

You still have the chance to improve

Projects

- Start working early and DO NOT procrastinate
- Do not make assumptions, if it's not clear, ask.
- Code Skeleton is mandatory
- Grading is NOT only about the output
- Make sure to correctly turnin projects, we post the list of students who don't have a submitted project. Check it.
- We might not accept un-submitted projects

Office hours

- Prepare your questions in advance
- Try to meet your GTA during the official office hours.
- Due to class size GTAs might not be able to see you individually
- GTA will help you troubleshoot the issue but cannot fix your code
- Example of unacceptable questions:
  - Emailing your entire code
  - Seeking assistance without trying by yourself first

Piazza

- We want to hear about your issues and get feedback, a good piazza post is:

subjective, states the facts, addresses the shortcomings, proposes solutions with a positive tone.

- Notice that post authors are not anonymous to class instructors

Attendance

- Missing 4 classes or 4 recitation will result in a penalty of 2.5%
- Ask your GTA about Recitation attendance
- Ask Prof. Rego about class attendance

Grading

- Redirect your email to the right person (not course instructor or coordinate):
- Lab grade: UTA
- Project grade: the posted GTA

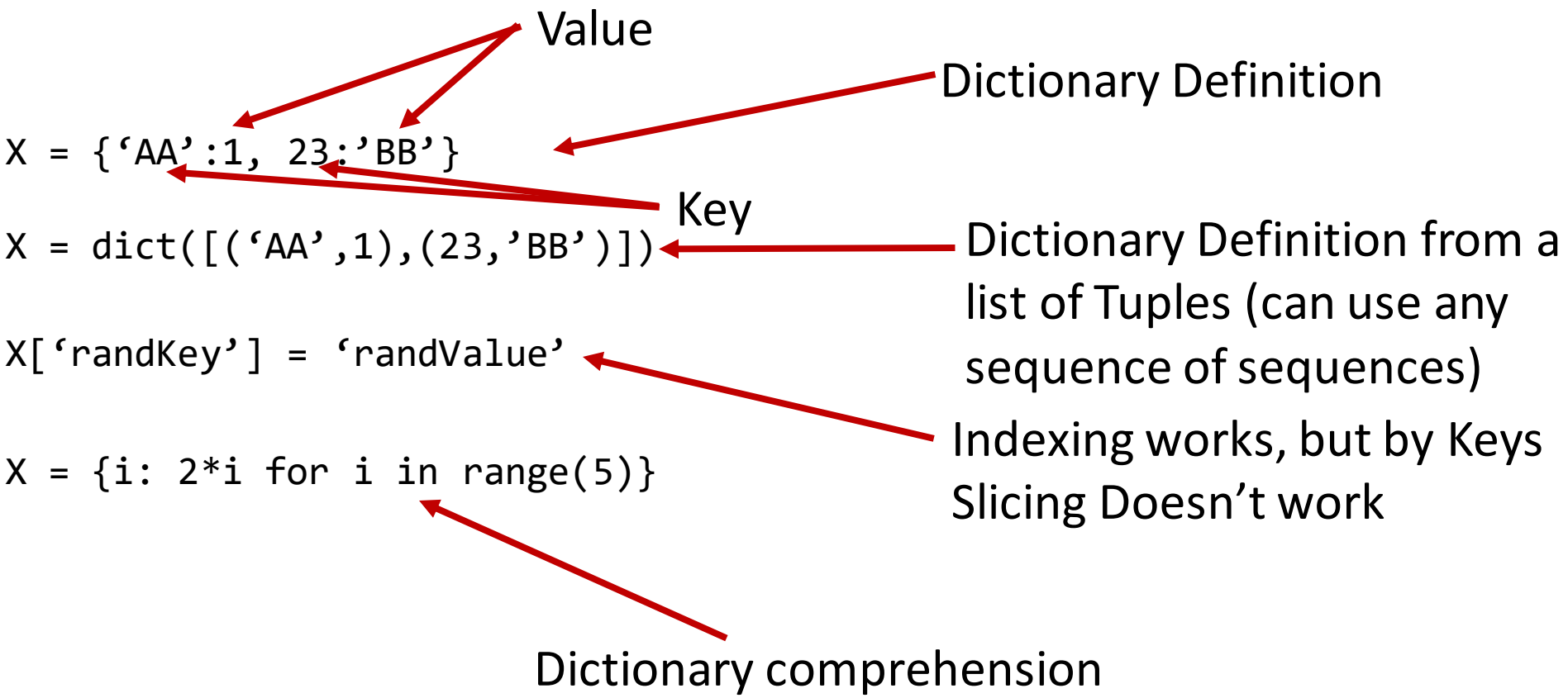
Finally …

- Learning how to code is a very useful skill
- Programming skills are gained by a lot practice
- Make good use of your lab, recitation and office hours time
- CS177 is a 4 credit hours class, allocate sufficient time

Objectives

- Recap of Dictionaries
- Sets
- Random Numbers
- Unit Testing

Dictionaries – Recap



Examples

- Convert an English sentence to French
  - You are given an English to French dictionary as a python “dict” type
  - Don’t worry about Grammar

Examples

- Convert an English sentence to French
  - You are given an English to French dictionary as a python dict type
  - Don’t worry about Grammar

```
frenchDict = {"I":"je", "am":"suis", "love":"aimer",
"very":"très", "much":"beaucoup"}

frenchDict[“happy”]=“heureux”

english = "I love python very much . I am very happy"

for englishWord in english.split():
    if (englishWord in frenchDict):
        print(frenchDict[englishWord], end=" ")
    else:
        print(englishWord, end=" ")
```

Examples

- Convert an English sentence to French
  - You are given an English to French dictionary as a python dict type
  - Don’t worry about Grammar

```
frenchDict = {"I":"je", "am":"suis", "love":"aimer",
"very":"très", "much":"beaucoup"}

frenchDict[“happy”]=“heureux”

english = "I love python very much . I am very happy"

for englishWord in english.split():
    if (englishWord in frenchDict):
        print(frenchDict[englishWord], end=" ")
    else:
        print(englishWord, end=" ")
```

Output:

je aimer python très beaucoup . je suis très heureux

Sets

- A collection of **unordered** and non duplicated values is called a **Set**
- Follow the abstract mathematical concept of a set
  - A collection of unique values
- Common use cases are membership testing, removing duplicates, set operations such as intersection and union etc

Sets– Define and Use

```
X = {2,3,4}

X = set({})

###X['randKey']
```

Set Definition
Empty Set must defined with the constructor, {} defines a dictionary

Unordered, and slicing and indexing both do not work

```
X = {2,3,4,3}
print(X)
{2,3,4}
```

Unique Values

Set Operations

- Length:

```
len({12, 5})
```

```
2
```

- Membership:

```
X = 2 in {2,3}
print(X)
```

```
True
```

- Iteration:

```
for x in {1,2,3}:
    print(x)
```

```
1
2
3
```

- Set Containment

```
{1}.issubset({1,2,3})
{1,2,3}.issuperset({1})
```

```
True
True
```

Set Operations

Sequence methods that employ order don't work on sets, like sort(), reverse() etc,

- <set>.clear(): Empty the Set

```
A={1,2}                                    {}
A.clear()
```

- <set>.add(x): Adds x to the Set

```
A={1,3}                                    {1,2,3}
A.add(2)
```

- <set>.remove(x): Removes x if present, raises KeyError Otherwise

```
A={1,3}                                    {1}
A.remove(3)
```

- <set>.discard(x): Removes x if present

```
A={1,3}                                    {1}
A.discard(3)
```

Set Operations

- <set1>.update(<set2>): Adds set2 entries to set1

```
A={1,3}                                    {1,3,5}
A.update({5})
```

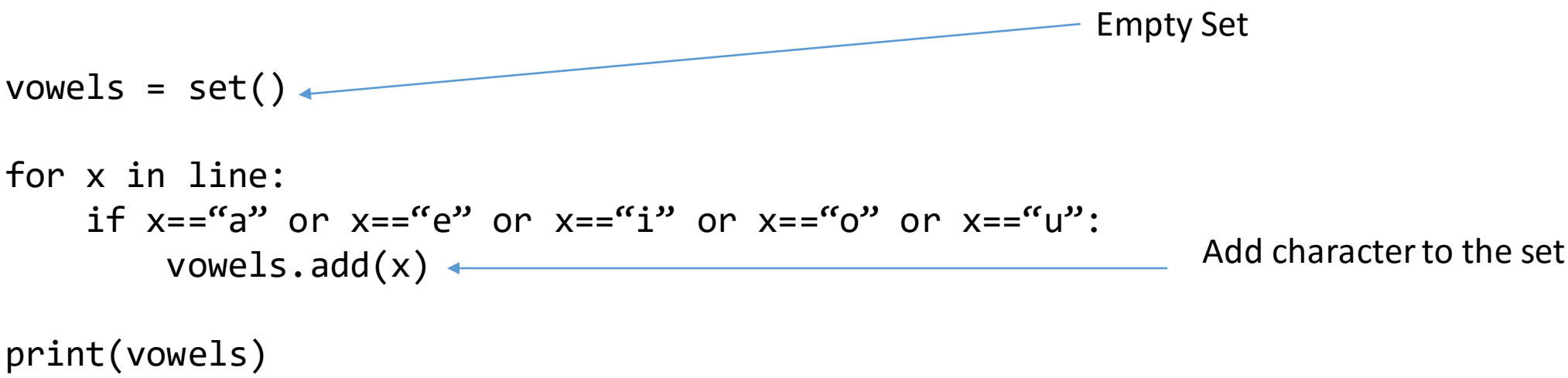
- Set Theory Operations
  - <set1>.intersection(<set2>): New Set with elements common to both sets
  - <set1>.union(<set2>): New Set with elements from both sets
  - <set1>.difference(<set2>): New Set with elements in set1 but not in set2
  - <set1>.symmetric_difference(<set2>): New Set with elements in either set1 or set2, but not in both

Examples

- Given a String line, find all the vowels in the string
  - Let us implement this with Sets

Examples

- Given a String line, find all the vowels in the string
  - Let us implement this with Sets
- Create a set of all the vowels in line.



Examples

- Given a String line, find all the vowels in the string
  - Let us implement this with Sets
- Create a set of all the vowels in line.

```
vowels = set()

for x in line:
    if x==“a” or x==“e” or x==“i” or x==“o” or x==“u”:
        vowels.add(x)

print(vowels)
```

Output:

Line: “abcdef”
Vowels: {“a”,”e”}

Line: “aabcd”
Vowels: {“a”}

Random Library

- Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc.
- The random module provides access to functions that support these types of operations. The random module is another library of functions that can extend the basic features of python.

Integer Random Number

- If we wanted a random integer, we can use the **randint** function.
- **Randint** accepts two parameters: a **lowest** and a **highest** number.
- For example:

```
import random

b = random.randint(0,5)

print("Generated random number is", b)
```

Random Choice

- **random.choice** function is used in order to Generate a random value from a sequence.
- For example:

```
import random

list1 = ['red','green','blue','brown']

print(random.choice(list1))
```

Shuffle function

- The shuffle function, shuffles the elements in list in place, so they are in a random order.
- Syntax: random.shuffle(list)

Shuffling example

```
from random import shuffle
list1 = [[i] for i range(10)]
print("Before shuffling", list1)
shuffle(list1)
print("After shuffling", list1)
```

```
>>>
Before shuffling [[0], [1], [2], [3], [4], [5], [6], [7], [8], [9]]
After shuffling [[9], [4], [5], [0], [2], [7], [3], [1], [8], [6]]
>>>
```

Randrange function

- Random.randrange function is used to generate a randomly selected element from range(start, stop, step).
- For example:

```
import random

for i in range(10):
      print(random.randrange(0,100,5))
```

Randrange example output

```
>>> ========================================== RESTART ==========================================
>>>
95
15
70
40
95
45
35
80
20
10
>>>
```

Flip a Coin Example

```
import random
def coinToss(number):
    heads = 0
    tails = 0
    for i in range(number): #range has start defaulted to 0
        flip = random.randint(1,2)
        if(flip == 1):
            heads = heads + 1
        else:
            tails = tails + 1
    return heads,tails
heads, tails = coinToss(10)
print("Number of heads:",heads)
print("Number of tails:",tails)
```

Some possible outputs

```
>>>
Number of heads: 5
Number of tails: 5
>>> ================================ RESTART ================================
>>>
Number of heads: 6
Number of tails: 4
>>> ================================ RESTART ================================
>>>
Number of heads: 4
Number of tails: 6
>>> ================================ RESTART ================================
>>>
Number of heads: 7
Number of tails: 3
>>>
```

Unit Testing

- In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use
- Intuitively, one can view a unit as the smallest testable part of an application
- a unit could be an entire program, but it is more commonly an individual function

Unit Testing Example

- Consider the program that we wrote for Project-1. It consisted of 2 functions the computeValue() function and the main() function. In order to test the implementation of our program is correct and we have not introduced any errors, it is always a good practice to test the implementation of our program at the lowest levels of the structure.
- So for our project1 program to work correctly, we need to test its individual components separately. So we will test the computeValue() function separately. We do this by importing our entire project-1 program and execute the computeValue() function as shown in the next slide.

Unit Testing Example

```
>>> import project1
>>> project1.computeValue (“New”, True, True, True)
13
>>> project1.computeValue (“Used”, True, False, True)
11
 >>> project1.computeValue (“New”, True, False, True)
12
>>> project1.computeValue (“Used”, True, True, True)
 12
```

Unit Testing Example

```
def computeValue(condition, gps, wifi, camera):
    # Declare and initialize variable (integer) to accumulate value of the phone
    value = 0
    # Increment value of variable based on its features
    if (condition == "New"):
        value += 10
    if (condition == "Used"):
        value += 9
    if (camera== True):
        value += 1
    if (wifi== True):
        value += 1
    if (gps== True):
        value += 1
    return value
```

Code for computeValue() function of Project-1

Unit Testing Example

- Notice that the computeValue() function takes 4 arguments and they are:
- Condition-It can take either the value of “New” or “Used”
- Gps-It can take either the value of “True” or “False”
- Wifi-It can take either the value of “True” or “False”
- Camera-It can take either the value of “True” or “False”
- So its possible to test the computeValue() function for 16 possible different inputs

Exercise-1

- Try to execute all the possible 16 unit test case scenarios for the computeValue() function which you wrote for Project-1

Exercise-2

- How would you do unit testing for the program that you wrote for Lab-05?

QUESTIONS?