National University of Computer and Emerging Sciences

Lab Manual 03 CL461-Artificial Intelligence Lab

Course Instructor	Sir Mubasher Baig
Lab Instructor (s)	Nimra Saima Akhtar
Section	H
Semester	Spring 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Table of Contents

1 Objectives

After performing this lab, students shall be able to understand Python data structures which include:

- Python Collections Review (Dictionary, List)
- Python Sets
- Python Exception Handling

2 Task Distribution

Total Time	**160 Minutes**
Demo	30 Minutes
Exercise	100 Minutes
Submission	30 Minutes

3 Python Sets

Sets have following characteristics:

- Set in Python is a data structure equivalent to sets in mathematics.
- Sets are a mutable collection of distinct (unique) immutable values that are unordered.
- Any immutable data type can be an element of a set: a number, a string, a tuple.
- Mutable (changeable) data types cannot be elements of the set.
- In particular, list cannot be an element of a set (but tuple can), and another set cannot be an element of a set.
- You can perform standard operations on sets (union, intersection, difference).

3.1 Set Initialization Examples

You can initialize a set in the following ways:

```
# Initialize empty set emptySet
= set()

# Pass a list to set() to initialize it
dataScientist = set(['Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'])
dataEngineer = set(['Python', 'Java', 'Scala', 'Git', 'SQL', 'Hadoop']
)
```

```
# Direct initialization using curly braces
dataScientist = {'Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'}
dataEngineer = {'Python', 'Java', 'Scala', 'Git', 'SQL', 'Hadoop'}
# Curly braces can only be used to initialize a set containing values
emptyDict= {}
```

3.2 Set Modification Examples

Let's consider the following set for our add/remove examples:

```
# Initialize set with values
graphicDesigner = {'InDesign', 'Photoshop', 'Acrobat', 'Premiere', 'Br
idge'}

# Add a new immutable element to the set
graphicDesigner.add('Illustrator')

# TypeError: unhasable type 'list'
graphicDesigner.add(['Powerpoint', 'Blender'])

# Remove an element from the set graphicDesigner.remove('Illustrator')

# Another way to remove an element. What is the difference?
graphicDesigner.discard('Premiere')

# Remove and return an arbitrary value from a set
graphicDesigner.pop()

# Remove all values from the set
graphicDesigner.clear()
```

3.3 Set Operations

Python sets have methods that allow you to perform these mathematical operations like union, intersection, difference, and symmetric difference. Let's initialize two sets to work on our examples:

```
# Initialize sets
dataScientist = set(['Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'])
dataEngineer = set(['Python', 'Java', 'Scala', 'Git', 'SQL', 'Hadoop']
)

# set built-in function union
dataScientist.union(dataEngineer)

# Equivalent Result
dataScientist | dataEngineer

# Intersection operation
dataScientist.intersection(dataEngineer)

# Equivalent Result
dataScientist & dataEngineer

# These sets have elements in common so isdisjoint would return False
dataScientist.isdisjoint(dataEngineer)

# Difference Operation
dataScientist.difference(dataEngineer)

# Equivalent Result
dataScientist - dataEngineer

# Symmetric Difference Operation
dataScientist.symmetric_difference(dataEngineer)

# Equivalent Result
dataScientist ^ dataEngineer
```

3.4 Frozensets

You have encountered nested lists and tuples. The problem with nested sets is that you cannot normally have nested sets as sets cannot contain mutable values including sets.

- A frozenset is very similar to a set except that a frozenset is immutable.
- The primary reason to use them is to write clearer, functional code.
- By defining a variable as a frozen set, you're telling future readers: do not modify this.
- If you want to use a frozen set you'll have to use the function to construct it. No other way.

```
# Nested Lists and Tuples
nestedLists = [['the', 12], ['to', 11], ['of', 9], ['and', 7], ['that'
, 6]]
nestedTuples = (('the', 12), ('to', 11), ('of', 9), ('and', 7), ('that
', 6))

# Initialize a frozenset
immutableSet = frozenset()

# Initialize a frozenset nestedSets
= set([frozenset()])
```

A major disadvantage of a frozenset is that since they are immutable, it means that you cannot add or remove values.

```
# AttributeError: 'frozenset' object has no attribute 'add'
immutableSet.add("Strasbourg")
```

4 Python Exception Handling

An exception is an error that is thrown by our code when the execution of the code results in an unexpected outcome. Normally, an exception will have an error type and an error message. Some examples are as follows.

```
ZeroDivisionError: division by zero
TypeError: must be str, not int
```

ZeroDivisionError and TypeError are the error type and the text that comes after the colon is the error message. The error message usually describes the error type.

4.1 Types of Exceptions

Here's a list of the common exceptions you'll come across in Python:

1. **ImportError**: It is raised when you try to import the library that is not installed or you have provided the wrong name
2. **IndexError:** Raised when an index is not found in a sequence. For example, if the length of the list is 10 and you are trying to access the 11th index from that list, then you will get this error
3. **IndentationError**: Raised when indentation is not specified properly
4. **ZeroDivisionError**: It is raised when you try to divide a number by zero
5. **ValueError**: Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified

6. **Exception**: Base class for all exceptions. If you are not sure about which exception may occur, you can use the base class. It will handle all of them.

4.2 Exception Handling with Try Except Clause

Python provides us with the try except clause to handle exceptions that might be raised by our code. The basic anatomy of the try except clause is as follows:

```
try:
    // some code
except:
    // what to do when the code in try raises an exception
```

In plain English, the try except clause is basically saying, "Try to do this, except (otherwise) if there's an error, then do this instead".

There are a few options on what to do with the thrown exception from the `try` block. Let's discuss them.

4.3 Re-raise the exception

Let's take a look at how to write the try except statement to handle an exception by re-raising it. First, let's define a function that takes two input arguments and returns their sum.

```
def myfunction(a, b):
    return a + b
```

Next, let's wrap it in a try except clause and pass input arguments with the wrong type so the function will raise the TypeError exception.

```
try:
    myfunction(100, "one
hundred") except:
print('error')     raise

  Output:
   raiseTraceback (most recent call
last):   File "<input>", line 2, in
<module>
  File "<input>", line 2, in myfunction
  TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

4.4 Catch certain types of exception

Another option is to define which exception types we want to catch specifically. To do this, we need to add the exception type to the `except` block.

```
try:
    myfunction(100, "one hundred")
except TypeError:
      print("Cannot sum the variables. Please pass numbers
only.")
Output:
Cannot sum the variables. Please pass numbers only.
```

To make it even better, we can actually log or print the exception itself.

```
try:
    myfunction(100, "one hundred")
except TypeError as e:
     print(f"Cannot sum the variables. The exception was:
{e}")
Output:
Cannot sum the variables. The exception was: unsupported
operand type(s) for +: 'int' and 'str'
```

Furthermore, we can catch multiple exception types in one `except` clause if we want to handle those exception types the same way. Let's pass an undefined variable to our function so that it will raise the `NameError`. We will also modify the `except` block to catch both `TypeError` and `NameError` and process either exception type the same way.

```
try:
    myfunction(100, a) except
(TypeError, NameError) as e:
     print(f"Cannot sum the variables. The exception was
{e}")
Output:
Cannot sum the variables. The exception was name 'a' is not
defined
```

4.5 Try….Finally

So far the try statement had always been paired with except clauses. But there is another way to use it as well. The try statement can be followed by a **finally** clause. Finally clauses are called

clean-up or termination clauses, because they must be executed under all circumstances, i.e. a "finally" clause is always executed regardless if an exception occurred in a try block or not. A simple example to demonstrate the finally clause:

```
try:
    x = float(input("Your number:
"))     inverse = 1.0 / x finally:
    print("There may or may not have been an
exception.") print("The inverse: ", inverse)
```

```
Your number: 34
There may or may not have been an exception.
The inverse:  0.029411764705882353
```

4.6 Try..except and finally

"finally" and "except" can be used together for the same try block, as it can be seen in the following Python example:

```
try:
    x = float(input("Your number:
"))     inverse = 1.0 / x except
ValueError:
    print("You should have given either an int or a float")
except ZeroDivisionError:
    print("Infinity")
finally:
    print("There may or may not have been an exception.")
```

```
Your number: 23 There may or may not have
been an exception.
```

Exercise (50 Marks)

Attempt all the questions below.

1: sets (30+10 marks)

Q1: Write a Python program to find the length of a set, apply all sets operations(union,intersection) and print the results,find maximum and the minimum value in a set,create a shallow copy of sets,check if a set is a subset of another set, remove all elements from a given set,check if two given sets have no elements in common.check if a given set is superset of itself and superset of another given set.

```
set1 = {1,2,3,4,5}
set2 = {4,5,6,7,8}
```

Q2:Python program to count the number of vowels using sets in a given string.

```
sample output

Input : Hello World
Output : No. of vowels :  3
```

2: Exception Handling (10 marks)

Q:3 Write a function to add, mul, divide two numbers x and y. Implement exception handling technique
(try..except clause) for handling possible exceptions in the scenario.