Shawn Louis Roll No.: 31

Experiment No: 4

|  |  |
| --- | --- |
| **Topic:** | To write a program using map(), filter(), reduce(), lambda functions and also exception handling feature of Python. |
| **Prerequisite:** | Knowledge of some programming language like C, Java |
| **Mapping With COs:** | CSL405.2 |
| **Objective**: | ­ To understand what an exception is and how it differs from a syntax error  - Able to handle exceptions in your Python program using try, except, else and finally statements |
| **Outcome:** | ­ This will motivate you to write clean, readable and efficient code in Python  - Ability to write program which handles various exceptions. |
| **Bloom’s Taxonomy** | Apply |
| **Theory/ Steps/ Algorithm/ Procedure:** | **filter() function**  The filter() function returns an iterator were the items are filtered through a function to test if the item is accepted or not.  Syntax: filter(function, iterable) |
|  | **lambda function** |
|  | A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression. |
|  | **Syntax**: lambda arguments : expression |
|  | The expression is executed and the result is returned |
|  | **map() function** |
|  | The map() function is similar to filter() function but it acts on each element of the sequence and perhaps change the elements. |
|  | **Syntax**: map(function, sequence) |
|  | **reduce() function** |

|  |  |
| --- | --- |
|  | The reduce() function reduces a sequence of elements to a single value by processing the elements according to a function supplied.  **Syntax**: reduce(function, sequence)  This is a part of functools module, so, include following line before using reduce()  **from functools import \*** |
| **Experiments:** | 1. Practice all the small exercises mentioned in the presentation discussed in class on “filter(), map(), reduce() functions” and “Exception Handling in Python”. 2. Write a Python program to find intersection of two given arrays using filter and Lambda. 3. Write a Python program to add two given lists using map and lambda. 4. Write a program (of your choice) using reduce() function to handle anyone exception in it. 5. The program below is not very robust. We can easily make it crash. Observe each function and see why it will fail the way it is called. Verify that you have made your functions more robust to erroneous input/data.   def example1():  for i in range( 3 ):  x = int( input( "enter a number: " ) )  y = int( input( "enter another number: " ) ) print( x, '/', y, '=', x/y )  def example2( L ):  print( "\n\nExample 2" ) sum = 0  sumOfPairs = []  for i in range( len( L ) ): sumOfPairs.append( L[i]+L[i+1] )  print( "sumOfPairs = ", sumOfPairs ) def main():  example1()  L = [ 10, 3, 5, 6, 9, 3 ]  example2( L )  example2( [ 10, 3, 5, 6, "NA", 3 ] )  example3( [ 10, 3, 5, 6 ] ) main() |

|  |  |
| --- | --- |
|  | 6. Design your own application based on any domain which handles minimum 15 exceptions in your python code. |
| **Deliverables:** | **Class Problems:**  ages = [5, 12, 17, 18, 24, 32, 9, 43, 10]  def myFunc(x):  if x < 18:  return False  else:  return True  adults = filter(myFunc, ages)  for x in adults:  print(x)    # list of alphabets  alphabets = ['a', 'b', 'd', 'e', 'i', 'j', 'o','s','o','i']  # function that filters vowels  def filterVowels(alphabet):  vowels = ['a', 'e', 'i', 'o', 'u']  if(alphabet in vowels):  return True  else:  return False  filteredVowels = filter(filterVowels, alphabets)  print('The filtered vowels are:')  for vowel in filteredVowels:  print(vowel)    l = [1,10,2,9,6,99,20,100,11,33]  def iseven(num):  if num % 2 == 0:  return True  else:  return False  newlist = filter(iseven, l)  for i in newlist:  print(i)    List = [1, 'a', 0, False, True, '0']  filteredList = filter(None, List)  print('The filtered elements are:')  for element in filteredList:  print(element)    x = lambda a : a + 10  print(x(5))  adder = lambda x, y: x + y  print (adder (1, 2))  x="Computer Dept."  (lambda x : print(x))(x)  y = (lambda x : print(x))  y(x)  print(y("Computer Dept."))  # Initialize a list of numbers (odd  #& even) and need to filter out only the even numbers  #in it  a = [1,2,3,4,5,6,7,8,9,0]  even = list(filter(lambda x: x % 2 == 0, a))  print(even)  odd = list(filter(lambda x:x % 2 != 0, a))  print(odd)    **SUM OF n + nn +nnn:**  n1 = input("Enter n : ")  n2 = n1\*2  n3 = n1\*3  sum = int(n1) + int(n2) + int(n3)  print(n1 + " + " + n2 + " + " + n3 + " = " + str(sum))    **Pig-Lating Translator:**  vowels = ('A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u')  word = []  def pigLatin(l):  for idx, i in enumerate(l):  if i[0] in vowels:  i += 'yay'  l[idx] = i  elif i[1] in vowels:  i += i[0] + 'ay'  i = i.replace(i[0], '', 1)  l[idx] = i  return(l)  def stringSplitter(s):  global word  w = ''  for i in s:  if i == " ":  word.append(w)  w = ''  continue  w += i  s = input("Enter string : \n")  stringSplitter(s)  word = pigLatin(word)  print("\nTranslated to Pig-Latin : ")  for i in word:  print(i, end = ' ')    **#Picks out items in a sequence (use range from -5 to 5) that are less than zero**  print(list(filter(lambda x: x < 0, range(-5, 5))))  print(list(filter(lambda x: x>0, range(-5,5))))  **#I have a list (iterable) of my favourite pet names, all in lower case and I need them in uppercase.**  **(try this without and with map function)**  pets = ['tom', 'jerry', 'husky']  for i in pets:  print(i.upper())  print(list(map(lambda x: x.upper(), ['tom', 'jerry', 'husky'])))  #**Initialize a list (iterable) of the scores of 10 students in a Maths exam. Let's filter out those who passed with scores more than 75...using filter**  marks = [10, 20, 40, 50, 30, 80, 60, 70, 90, 100]  print(list(filter(lambda x: x > 75, marks)))  **Write a python code for palindrome detector. Let's filter out words that are palindromes from a tuple (iterable) of suspected palindromes.**  words = ['mom', 'bro', 'dad', 'sis']  print(list(filter(lambda x: x == x[::-1], words)))    **Task 2 : Write a Python program to find intersection of two given arrays using filter and Lambda**.  a = list(map(int, input("Enter list 1 : ").split()))  b = list(map(int, input("Enter list 2 : ").split()))  print("Intersection of the 2 lists : ", list(filter(lambda x: x in a, b)))    **Task 3:**  **Write a Python program to add two given lists using map and lambda.**    a = list(map(int, input("Enter list 1 : ").split()))  b = list(map(int, input("Enter list 2 : ").split()))  print(“Addition : “, list(map(lambda x,y: x+y, a, b)))    **Task 4 :**  **Write a program (of your choice) using reduce() function to handle anyone exception in it**  from functools import reduce  a = list(map(int, input("Enter list : ").split()))  print(“Sum is : “, reduce(lambda x,y: x+y, a))    **Task 5 :  The program below is not very robust. We can easily make it crash. Observe each function and see why it will fail the way it is called. Verify that you have made your functions more robust to erroneous input/data.**  def example1():  for i in range( 3 ):  x = int( input( "\nenter a number : " ) )  y = int( input( "enter another number : " ) )  print( x, '/', y, '=', x/y )  def example2( L ):  print("\n\nExample 2" )  sum = 0  sumOfPairs = []  for i in range( len( L ) ):  if i == len(L) - 1:  sumOfPairs.append(L[i] + L[1])  else:  sumOfPairs.append( L[i]+L[i+1] )  print( "sumOfPairs = ", sumOfPairs )  def main():  while True:  try:  example1()  L = [ 10, 3, 5, 6, 9, 3 ]  example2( L )  example2( [ 10, 3, 5, 6, 3 ] )  example3( [ 10, 3, 5, 6 ] )  except ZeroDivisionError:  print("Please enter a non-zero number!")  except ValueError:  print("Please enter an integer value!")  except IndexError:  print("Array bounds out of range")  except TypeError:  print("Cannot add str and int datatypes")  except NameError:  print("Undefined variable or function")  else:  print("Everything's fine!")  break;  main()      **Task 6:**  **Design your own application based on any domain which handles minimum 15 exceptions in your**  **python code.**  import math  import sys  try:  x = int(input('Please enter a positive number:\n'))  try:  print(f'Square Root of {x} is {math.sqrt(x)}')  except ValueError as ve:  print(f'You entered {x}, which is not a positive number.')  y = int(input('Please enter a positive number:\n'))  try:  try:  print("DIVISION: "+(x/y))  except TypeError as ve:  print(f'Value Error!')  except ZeroDivisionError as ve:  print(f'You entered {y}, which is 0.')  try:  print("Calculation Complete!");  except IndentationError as ve:  print("There was a syntax error!")  try:  abc  except NameError as ve:  print("Name Error!")  assert y != 0, "Invalid Operation"  print(x / y)  try:  X = 10  X.append(5)  except AttributeError as ve:  print("Attribute Error!")  try:  a = 10 / 0  print(a)  except ArithmeticError:  print("This statement is raising an arithmetic exception.")  else:  print("Success.")  try:  n = int(v)  except Exception:  print("Couldn't parse")  try:  a = [5, 8, 17]  print(a[17])  except LookupError:  print("Index out of bound error.")  else:  print("Success")  try:  print(math.exp(1000))  except OverflowError as ve:  print("Overflow Error!")  try:  import module\_does\_not\_exist  except ImportError as ve:  print("Import Error!")  except ValueError as ve:  print('You are supposed to enter positive number.')  try:  f = open('myfile.txt')  s = f.readline()  i = int(s.strip())  except OSError as err:  print("OS error: {0}".format(err))  try:  my\_list = [5,6, 8, 4, 17,5]  print(my\_list[6])  except IndexError as e:  print(e)  try:  spec.loader.exec\_module(module)  except BaseException:  try:  del sys.modules[spec.name]  except KeyError:  pass  try:  f = open('myfile.txt')  s = f.readline()  i = int(s.strip())  except IOError as e:  print ("I/O error({0}): {1}".format(e.errno, e.strerror))  except ValueError:  print ("Could not convert data to an integer.")  except:  print ("Unexpected error:", sys.exc\_info()[0])  **Output:** |
| **Conclusion:** | Thus we have successfully able to write a program which handles various exceptions. |
| **References:** | <https://www.w3schools.com/python/python_try_except.asp> <https://www.programiz.com/python-programming/exception-handling> |

Don Bosco Institute of Technology Department of Computer Engineering

Academic year – 2019­20 Open Source Technology Lab

Assessment Rubric for Experiment No.: 4

Performance Date : Submission Date :

**Title of Experiment** : map(), filter(), reduce(), lambda functions and also

exception handling feature of Python

**Year and Semester** : 2nd Year and IVth Semester

**Batch** : Computer

Name of Student : Shawn Louis

Roll No. : 31

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance** | **Poor** | **Satisfactory** | **Good** | **Excellent** | **Total** |
| 2 points | 3 points | 4 points | 5 points |  |
| **Results and Documentati ons** | **Poor** | **Satisfactory** | **Good** | **Excellent** |
| 2 points | 3 points | 4 points | 5 points |
| **Timely Submission** | **Submissio n beyond 14 days of the deadline** | **Late submission till 14 days** | **Late submission till 7 days** | **Submission on time** |
| 2 points | 3 points | 4 points | 5 points |

Signature

(Sana Shaikh)