**ASSIGNMENT – 04**

**Q. Write a program to replace each string with an integer value in a given list of strings.**

**The replacement integer value should be a sum of AScci values of each character of type**

**corresponding string........**

**LIST: ['GAnga', 'Tapti', 'Kaveri', 'Yamuna', 'Narmada' ]**

list\_rivers =['GAnga', 'Tapti', 'Kaveri', 'Yamuna', 'Narmada' ]

river = [] // empty list

for i in list\_rivers:

num=0

for j in i:

num = num + ord(j)

river.append(num) // [597, 813, 1143, 692, 619, 787, 610, 514]

**Q. You have to run your Program at 9:00am. Date: 14th April 2020.**

from datetime import datetime

today = datetime.now()

print(today)

**OUTPUT:**

2020-04-13 17:11:01.952975

**Q. GIve a tuple:**

**tuple = ('a','l','g','o','r','i','t','h','m')**

1. **Using the concept of slicing, print the whole tuple.**

my\_tuple = ('a','l','g','o','r','i','t','h','m')

print(my\_tuple[1:4])

print(my\_tuple[:-7])

print(my\_tuple[7:])

print(my\_tuple[:])

**OUTPUT:**

('l', 'g', 'o')

('a', 'l')

('h', 'm')

('a', 'l', 'g', 'o', 'r', 'i', 't', 'h', 'm')

1. **delete the element at the 3rd Index, print the tuple.**

tuplex = ('a', 'l', 'g', 'o', 'r', 'i', 't', 'h', 'm')

print(tuplex)

tuplex = tuplex[:2] + tuplex[3:]

print(tuplex)

listx = list(tuplex)

listx.remove("o")

tuplex = tuple(listx)

print(tuplex)

**OUTPUT:**

('a', 'l', 'g', 'o', 'r', 'i', 't', 'h', 'm')

('a', 'l', 'o', 'r', 'i', 't', 'h', 'm')

('a', 'l', 'r', 'i', 't', 'h', 'm')

**Q. Take a list REGex=[1,2,3,4,5,6,7,8,9,0,77,44,15,33,65,89,12]**

1. **print only those numbers greator then 20**

num =[1,2,3,4,5,6,7,8,9,0,77,44,15,33,65,89,12]

print()

print(all(x > 20 for x in num))

print()

**OUTPUT:**

[77, 44,33,65,89]

1. **then print those numbers those are less then 10 or equal to 10**

num =[1,2,3,4,5,6,7,8,9,0,77,44,15,33,65,89,12]

print()

print(all(x < 10 for x in num))

print(all(x = 10 for x in num))

print()

**OUTPUT:**

[1,2,3,4,5,6,7,8,9,0]

1. **store these above two list in two different list.**

def merge(list1, list2):

merged\_list = [(list1[i], list2[i]) for i in range(0, len(list1))]

return merged\_list

list1 = [1, 2, 3]

list2 = ['a', 'b', 'c']

print(merge(list1, list2))

**Q. Execute standard LINUX Commands using Python Programming.**

You can execute the command in a subshell using os.system(). This will call the Standard C function system(). This function will return the exit status of the process or command. This method is considered as old and not recommended, but presented here for historical reasons only. The subprocess module is recommended and it provides more powerful facilities for running command and retrieving their results.

**ASSIGNMENT – 05**

**Q. Difference between Return and Yield ?**

The **yield** statement suspends function’s execution and sends a value back to the caller, but retains enough state to enable function to resume where it is left off. When resumed, the function continues execution immediately after the last yield run. This allows its code to produce a series of values over time, rather than computing them at once and sending them back like a list.

Let’s see with an example:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| def simpleGeneratorFun():      yield 1      yield 2      yield 3  for value in simpleGeneratorFun():      print(value) |

**OUTPUT:**

1

2

3

**Return** sends a specified value back to its caller whereas Yield can produce a sequence of values. We should use yield when we want to iterate over a sequence, but don’t want to store the entire sequence in memory.

Yield are used in Python generators. A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| def nextSquare():      i = 1;      while True:          yield i\*i          i += 1  # Next execution resumes  for num in nextSquare():      if num > 100:           break      print(num) |

**OUTPUT:**

1

4

9

16

25

36

49

64

81

100

**Q. Make digital Clock and run it for 5 sec.**

import time

while True:

from datetime import datetime

now = datetime.now()

print ("%s/%s/%s %s:%s:%s" % (now.month,now.day,now.year,now.hour,now.minute,now.second))

time.sleep(1)

**OUTPUT:**

16:39:08

:09

:10

:11

:12

**Q. Add anything in tuple.. example: (1,2,3,4) -> new tuple (1,2,3,4,5)**

tuplex = (1,2,3,4)

print(tuplex)

tuplex = tuplex + (5,)

print(tuplex)

tuplex = tuplex[:5] + (6,)

print(tuplex)

tuplex = tuplex[:6] + (7,)

print(tuplex)

tuplex = tuplex[:7] + (8,)

print(tuplex)

listx = list(tuplex)

listx.append(30)

tuplex = tuple(listx)

print(tuplex)

**OUTPUT:**

(1, 2, 3, 4)

(1, 2, 3, 4, 5)

(1, 2, 3, 4, 5, 6)

(1, 2, 3, 4, 5, 6, 7)

(1, 2, 3, 4, 5, 6, 7, 8)

(1, 2, 3, 4, 5, 6, 7, 8, 30)

**Q. WhatsApp texting using webbrowser Lib.**

from selenium import webdriver

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

import time

driver = webdriver.Chrome('/home/saket/Downloads/chromedriver')

driver.get("https://web.whatsapp.com/")

wait = WebDriverWait(driver, 600)

target = '"Friend\'s Name"'

string = "Message sent using Python!!!"

x\_arg = '//span[contains(@title,' + target + ')]'

group\_title = wait.until(EC.presence\_of\_element\_located((

By.XPATH, x\_arg)))

group\_title.click()

inp\_xpath = '//div[@class="input"][@dir="auto"][@data-tab="1"]'

input\_box = wait.until(EC.presence\_of\_element\_located((

By.XPATH, inp\_xpath)))

for i in range(100):

input\_box.send\_keys(string + Keys.ENTER)

time.sleep(1)