Deep Learning Lab 1

Selvakumar G (22MAI1004)

Program 1 : Get the following input from the user: first name and last name. Write a Python program to print as

Hello first name last name   
Welcome to Python!

Code:

name = **input**("Enter your Full Name: ")

names = name.**split**(" ")

**print**(f"Hello {names[0]} {names[1]}\n Welcome to Python!")

Output:

Text

Description automatically generated

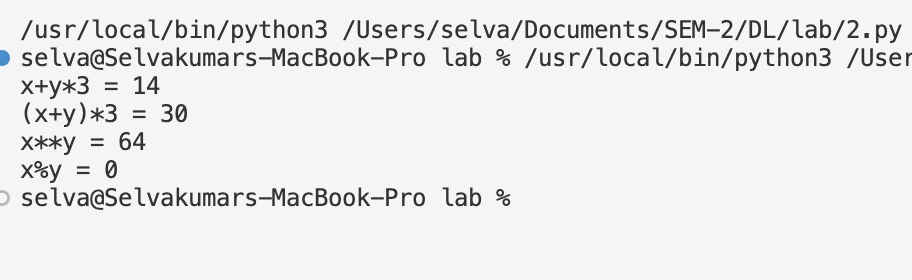
Program 2: Assign x = 8 and y = 2. Evaluate the following expressions and write the output,   
a) x + y \* 3   
b) (x + y) \* 3   
c) x \*\* y   
d) x % y

Code:

x, y = 8, 2

**print**(f"x+y\*3 = {x+y\*3}\n(x+y)\*3 = {(x+y)\*3}\nx\*\*y = {x\*\*y}\nx%y = {x%y}")

Output:



Program 3: An employee’s total weekly pay equals the hourly wage multiplied by the total   
number of regular hours plus any overtime pay. Overtime pay equals the total   
overtime hours multiplied by 1.5 times the hourly wage. Write a program that takes as   
inputs the hourly wage, total regular hours, and total overtime hours and displays an   
employee’s total weekly pay.

Code:

hourly\_wage = **float**(**input**("Enter the Hourly Wage of the Employ: "))

total\_regular\_hours = **float**(**input**("Enter the total regular Hours: "))

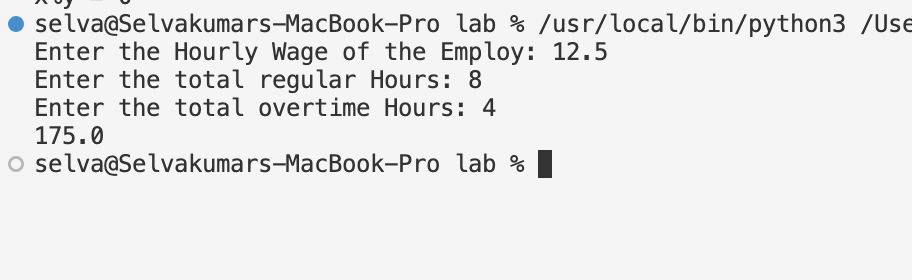
total\_overtime\_hours = **float**(**input**("Enter the total overtime Hours: "))

overtime\_pay = total\_overtime\_hours \* (1.5 \* hourly\_wage)

total\_weekly\_pay = hourly\_wage \* total\_regular\_hours + overtime\_pay

**print**(total\_weekly\_pay)

Output:



Problem 4: A company decides to give bonus to all its employees for Diwali. A 5% bonus on salary is given to the male workers and 15% bonus on salary to the female workers. If the salary of the employee is less than Rs. 10000/- then the employee gets an extra 2% bonus on salary. Calculate the bonus that the employee will get and display the total salary.

Code:

gender = **input**("Enter the Employee Gender (M for Male, F for Female) : ")

salary = **int**(**input**("Enter the Employee Salary"))

total\_salary = salary

if salary < 10000:

bonus = salary \* 0.02

total\_salary += bonus

if gender == 'm':

bonus = salary \* 0.05

total\_salary += bonus

elif gender == "f":

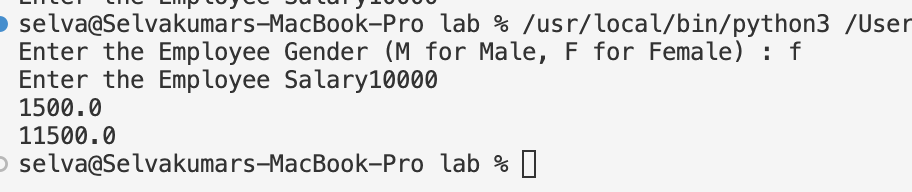
bonus = salary \* 0.15

total\_salary += bonus

**print**(bonus)

**print**(total\_salary)

Output:



Problem 5: Write a series of Python statements that will read three strings from the user, and then print them in dictionary order. (Note: you can compare two strings using the relational operators).

Code:

str1 = **input**("Enter a string: ")

str2 = **input**("Enter a string: ")

str3 = **input**("Enter a string: ")

if str1 < str2 and str1 < str3:

if str2 < str3:

**print**(str1, str2, str3)

else:

**print**(str1, str3, str2)

elif str2 < str1 and str2 < str3:

if str1 < str3:

**print**(str2, str1, str3)

else:

**print**(str2, str3, str1)

else:

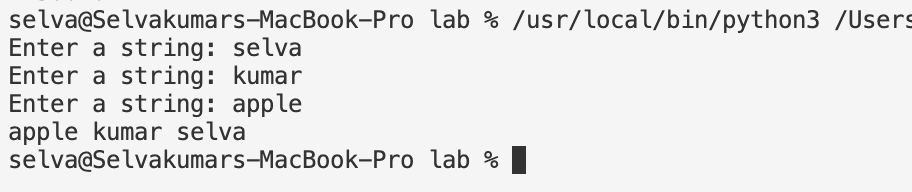
if str1 < str2:

**print**(str3, str1, str2)

else:

**print**(str3, str2, str1)

Output:



Program 6: Write a program that takes user’s name and PAN card number. Validate the information using string functions

Code:

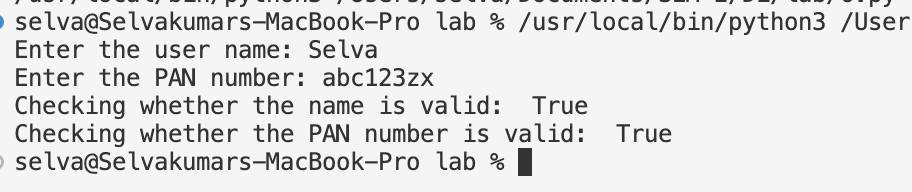
name = **input**("Enter the user name: ")

pan\_no = **input**("Enter the PAN number: ")

**print**("Checking whether the name is valid: ", name.**isalpha**())

**print**("Checking whether the PAN number is valid: ", pan\_no.**isalnum**())

Output:



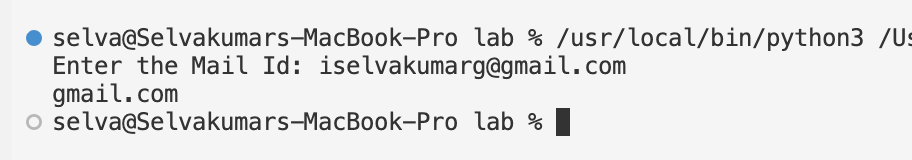
Program 7: Write a Python program to parse an email id to print from which email server it was sent.

Code:

mail = **input**("Enter the Mail Id: ")

**print**(mail.**split**("@")[1])

Output:



Program 8: Write a Python program to strip a set of characters from a string Encrypt a given message by “rotating” each letter by a fixed number of places. To rotate a letter means to shift it through the alphabet, wrapping around to the beginning if necessary, so ‘A’ rotated by 3 is ‘D’ and ‘Z’ rotated by 1 is ‘A’. Write a function called rotate\_word that takes a string and an integer as parameters, and returns a new string that contains the letters from the original string rotated by the given amount. E.g Given String: HAL Encrypted String: JCN (Rotated by 2)

Code:

def **rotate\_word**(s, n):

return ''.**join**([**chr**(**ord**(i) + n) for i in s])

**print**(**rotate\_word**('HAL', 2))

Output:

Text

Description automatically generated

Program 9: Ackermann Function

Code:

def **ack**(m, n):

if m == 0:

return n + 1

elif m > 0 and n == 0:

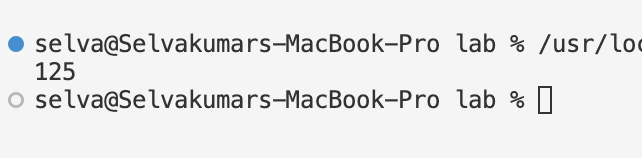
return **ack**(m - 1, 1)

elif m > 0 and n > 0:

return **ack**(m - 1, **ack**(m, n - 1))

**print**(**ack**(3, 4))

Output:



Program 10: Given a list, find frequency of each element and save it as list of tuple [(number, frequency)]. Input : test\_list = [4, 5, 4, 5, 6, 6, 5] Output : [(4, 2), (5, 3), (6, 2)] Input : test\_list = [4, 5, 4, 5, 6, 6, 6] Output : [(4, 2), (5, 3), (6, 3)]

Code:

def **frequency**(l):

return [(i, l.count(i)) for i in **set**(l)]

**print**(**frequency**([4, 5, 4, 5, 6, 6, 5]))

**print**(**frequency**([4, 5, 4, 5, 6, 6, 6]))

Output:

Text

Description automatically generated with medium confidence

Program 11: Remove duplicates from the tuple

Code:

def **remove\_duplicates**(t):

return **tuple**(**set**(t))

**print**(**remove\_duplicates**((1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)))

Output:

Text

Description automatically generated with medium confidence

Program 12: Give an appropriate list comprehension for each of the following: i) L1 = [1, ’x’, 4, 5.6, ’z’, 9, ‘a’, 0, 4] create a list which consists of integer values. ii) Producing a list of consonants that appear in string w. iii) Multiples of 10 (n values) iv) Construct a list of the form: [‘1a’,’2a’,’3a’,’4a’] v) Create a list which stores the sum of each of the elements from the two lists.

*# i) L1 = [1, ’x’, 4, 5.6, ’z’, 9, ‘a’, 0, 4] create a list which consists of integer values.*

L1 = [1, 'x', 4, 5.6, 'z', 9, 'a', 0, 4]

L2 = [i for i in L1 if **type**(i) == **int**]

**print**(L2)

*# ii) Producing a list of consonants that appear in string w.*

w = 'This is a string'

L3 = [i for i in w if i not in 'aeiou']

**print**(L3)

*# iii) Multiples of 10 (n values)*

L4 = [i for i in **range**(100) if i % 10 == 0]

**print**(L4)

*# iv) Construct a list of the form: [‘1a’,’2a’,’3a’,’4a’]*

L5 = [**str**(i) + 'a' for i in **range**(1, 5)]

**print**(L5)

*# v) Create a list which stores the sum of each of the elements from the two lists.*

L6 = [1, 2, 3, 4]

L7 = [5, 6, 7, 8]

L8 = [L6[i] + L7[i] for i in **range**(**len**(L6))]

**print**(L8)

Output

Text

Description automatically generated

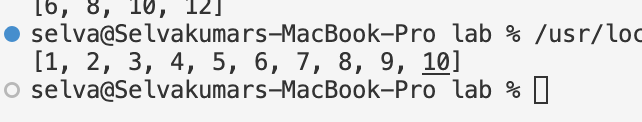
Problem 13: Obtain a nested list use looping constructs and list comprehension to flatten the list. Example: [[1],[2,3],[4,5,6],[7,8,9,10]] o/p: [1,2,3,4,5,6,7,8,9,10]

Code:

def **flatten\_list**(l):

return [i for j in l for i in j]

**print**(**flatten\_list**([[1],[2,3],[4,5,6],[7,8,9,10]]))

Output:  


Problem 14: Create a dictionary for 6 employee details with empno as key, name, dob and net-pay as list of values use appropriate dictionary methods: a. Create a dictionary with the above information. b. Insert a new employee details as the second employee c. Delete the employee at the 4th position d. Delete the last employee e. Increment the salary of all employees by 5%.

Code:

*#a. Create a dictionary with the above information.*

employee\_dict = {

101: ["A", "1990-11-25", 70000],

102: ["B", "1991-05-21", 10000],

103: ["C", "1992-11-03", 35000],

104: ["D", "1993-09-27", 85000],

105: ["E", "1984-06-28", 80000],

106: ["F", "1993-03-25", 15000]

}

*# b. Insert a new employee details as the second employee*

employee\_dict[102] = ["G", "1994-08-12", 50000]

*# C. Delete the employee at the 4th position*

employee\_dict.**pop**(104)

*# d. Delete the last employee*

employee\_dict.**popitem**()

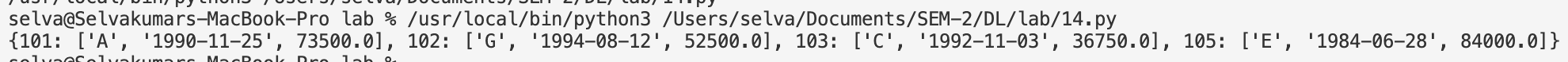
*# e. Increment the salary of all employees by 5%*

for emp\_details in employee\_dict. **values**():

emp\_details[2] \*= 1.05

**print**(employee\_dict,end="\n")

Output:



Problem 15: Consider the student details are maintained using nested dictionary as follows: {Reg no: {subcode: CAT1, CAT2, SAT}} a. Create nested dictionary for three subjects b. Display the information of a student given his register number c. To display the marks of a student given his subject code d. Update the details of the student given the register number.

Code:

students = {

"101": {

"sub1": [50, 60, 75],

"sub2": [70, 80, 90],

"sub3": [85, 90, 95]

},

"102": {

"sub1": [60, 75, 80],

"sub2": [80, 85, 90],

"sub3": [90, 95, 98]

},

"103": {

"subl": [70, 80, 85],

"sub2": [85, 90, 92],

"sub3": [95, 98, 99]

}

}

*#b. Displaying the information of a student given their register number:*

def **display\_student\_info**(reg\_no):

if reg\_no in students:

**print**(f"Register number: {reg\_no}")

for sub\_code, marks in students[reg\_no].**items**():

**print**(f"{sub\_code} marks: {marks}")

else:

**print**(f"No student found with register number {reg\_no}")

*# c. Displaying the marks of a student in a particular subject given their register number and subject code:*

def **display\_marks\_by\_sub\_code**(reg\_no, sub\_code):

if reg\_no in students and sub\_code in students[reg\_no]:

**print**(f"{sub\_code} marks for student {reg\_no}: {students[reg\_no][sub\_code]}")

else:

**print**(f"No marks found for subject {sub\_code} and student {reg\_no}")

*# d. Updating the details of a student given their register number, subject code and marks:*

def **update\_student\_details**(reg\_no, sub\_code, cat1, cat2, sat):

if reg\_no in students:

if sub\_code in students[reg\_no]:

students [reg\_no][sub\_code] = [cat1, cat2, sat]

**print**(f"Updated {sub\_code} marks for student {reg\_no}: {students[reg\_no][sub\_code]}")

**print**(f"No subject {sub\_code} found for student {reg\_no}")

else:

**print**(f"No student found with register number {reg\_no}")

**print**(**display\_student\_info**( "101"))

**print**(**display\_marks\_by\_sub\_code**("101", "sub1"))

**print**(**update\_student\_details**("101", "sub1", 70,70,70))

Output:

Text

Description automatically generated with medium confidence

Problem 16: Write a Python code to read the content of ‘ebook.txt’ and display the contents of the file onto the console.

ebook.txt

Text

Description automatically generated

Code:

with **open**("ebook.txt", "r") as f:

**print**(f.**read**())

Output:

Text, letter

Description automatically generated

Problem 17: Write a function ‘display\_words()’ in python to read lines from a text file "ebook.txt", and returns a list with words less than 4 characters.

Code:

def **display\_words**():

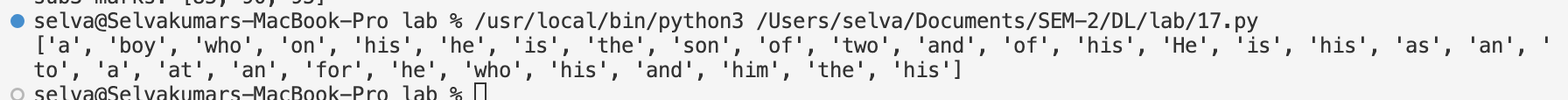
with **open**("ebook.txt", "r") as f:

words = f.**read**().**split**()

return [word for word in words if **len**(word) < 4]

**print**(**display\_words**())

Output:



Problem 18: Create a CSV file and store student details like Name, Register number and CGPA. Write a python code to read the file content and display the content. Duplicate entries and null value entries has to be ignored while displaying.

Code:

*# Create a CSV file and store student details like Name, Register number and CGPA. Write a python code to read the file content and display the content. Duplicate entries and null value entries has to be ignored while displaying.*

import **csv**

*# Sample data*

data = [

['Selva', '1001', '9.1'],

['Shiva', '1002', '9.1'],

['Nandhini', '1003', '7.8' ],

['Sri', '1004', '8.9' ],

['pawan', '1005' , '9.5'],

['Bavani', '1006', '8.5'],

['', '1007', '9.1'],

['shankar', '', '7.8' ]

]

with **open**('student.csv', mode='w', newline='') as file:

writer = **csv**.**writer**(file)

writer.**writerow**(['Name', 'Reg No', 'CGPA'])

for row in data:

if row[0] and row[1] and row[2]:

writer.**writerow**(row)

with **open**( 'student.csv', mode= 'r') as file:

reader = **csv**. **reader**(file)

**next**(reader)

for row in reader:

if **len**(row) == 3 and row[0] and row[1] and row[2]:

**print**(f"Name: {row[0]}, Reg No: {row[1]}, CGPA: {row[2]}")

Output:

Text

Description automatically generated