**SCHOOL OF COMPUTING**

**IT8701 Introduction to Programming for Data Science**

**Practical 1 Submission Worksheet (Graded as part of CA3)**

|  |
| --- |
| **Instructoins:**   1. Submit this at PoliteMall “Lab Submissions->Practical 1 Submission” folder 2. Name your file “YourClass-YourName-YourStudentID.docx” |

|  |  |
| --- | --- |
| **Name** | Muhammad Iylia Bin Mohd Hutta |
| **Student ID** | P7474841 |
| **Class** | NSDDA1/CE/2220/4 |

# Section 2 Tasks

### Task 4: Average of electricity bills

|  |
| --- |
| *from* statistics *import* mean  *# declare empty list*  bill6mth = []  *# for loop to get each month's bill into list*  *for* i *in* range(1,7):  bill6mth.append(float(input(f'Enter bill #{i}: ')))  *# border and header to match sample output*  border = '\*'  header = 'Calculate the average of your last 6-months electrcity bill'  print(border\*len(header))  print(header)  print(border\*len(header))  *# print bills for each month*  print(f'Your electricity bills for the past 6 months are: ${bill6mth[0]:.2f}, ${bill6mth[1]:.2f}, ${bill6mth[2]:.2f}, ${bill6mth[3]:.2f}. ${bill6mth[4]:.2f}, ${bill6mth[5]:.2f}')  *# print the average bill over 6 months*  print(f'The average of your electricity bill is ${mean(bill6mth):.2f}') |

|  |
| --- |
|  |

# Section 5 Tasks

### Task 2: Perform simple string operations

|  |
| --- |
| *# prompt the user to enter 3 random strings of at least 8 characters long, each separated by a comma*  random\_strings = input('Please enter 3 random strings of at least 8 characters long, each separated by a comma:')  *# split the random strings by comma separator*  s1 = random\_strings.split(',')[0]  s2 = random\_strings.split(',')[1]  s3 = random\_strings.split(',')[2]  *# calculate length of each random string*  len\_s1 = len(s1)  len\_s2 = len(s2)  len\_s3 = len(s3)  *# slice the strings*  second\_third\_s1 = s1[1:3]  fifth\_to\_seven\_s2 = s2[4:7]  last\_two\_s3 = s3[-2:]  print(f's1 is {s1}')  print(f'Length of {s1} is {len\_s1}')  print(f'2nd and 3rd characters of {s1} is {second\_third\_s1}\n')  print(f's2 is {s2}')  print(f'Length of {s2} is {len\_s2}')  print(f'5th to 7th characters of {s2} is {fifth\_to\_seven\_s2}\n')  print(f's3 is {s3}')  print(f'Length of {s3} is {len\_s3}')  print(f'Last two characters of {s3} is {last\_two\_s3}') |

|  |
| --- |
| Text  Description automatically generated |

# Section 6 Tasks

### Task 2: Spdonalds

|  |
| --- |
| *# define function to print price of chosen item, prompt for qty, calculate price and display subtotal*  def create\_order\_item(*menu\_key*, *menu*):  *# print price of item*  print(f"{*menu*[*menu\_key*]['item']} ${*menu*[*menu\_key*]['price']:.2f} added!")  *# prompt user for quantity*  quantity = int(input(f"How many {*menu*[*menu\_key*]['item']} do you want to order?:"))  *# calculate price with quantity and store*  subtotal = quantity\**menu*[*menu\_key*]['price']  *# display subtotal*  print(f"The total cost for {quantity} {*menu*[*menu\_key*]['item']} is ${subtotal:.2f}\n")  *return*[*menu\_key*, quantity, subtotal]  print('Welcome to SPdonalds!')  *# create dictionary for menu and prices*  menu = {  1: {'item': 'SPMuffin', 'price': 5.00},  2: {'item': 'SPancakes', 'price': 3.00},  3: {'item': 'SPHashbrown', 'price': 1.50}  }  print('Below is our Breakfast menu:')  *# print menu with keys and items and prices*  *for* key *in* menu:  print(f"{key}. {menu[key]['item']} (${menu[key]['price']:.2f})")  *# prompt for key to exit menu*  print('\nKey in [Q] to exit\n')  order = []  *while* True:  *# prompt user to key in choice of item*  choice = input('Enter your choice of food:')    *# key to quit program or when invalid choice*  *if* choice.upper() == 'Q':  print('Exiting program....')  *break*;  *elif* int(choice) > 3:  print('Sorry, you have entered an invalid choice\nUnable to continue. Exiting program....')  *break*;  *else*:  menu\_key = int(choice)  *# if choice in menu:*  *if* menu\_key >= 0 and menu\_key <= len(menu):  *# use function*  order\_item = create\_order\_item(menu\_key, menu) |

|  |
| --- |
| Text  Description automatically generated with low confidence |

# Section 7 Tasks

### Task 2: Calculate sum of numbers within a range

|  |
| --- |
| *# Inform the user the purpose of the program*  print('This program prints the sum of a range of numbers from x to y\n')  print('For example, if x is 10 and y is 50, the program will print the sum of numbers from 10 to 50\n')  *# Prompt the user to enter the value for x and y.*  *# check if 1) x and y are numeric*  *# 2) higher than zero*  *# 3) y is greater than x.*  *# If not, display an error message and terminate the program.*  x = input('Please enter the value of x:')  y = input('Please enter the value of y:')  *if* x.isnumeric() != True or y.isnumeric() != True:  print('One or more of your inputs are not numeric!\nUnable to continue. Program terminated.')  *elif* int(x) <= 0 or int(y) <= 0:  print('One or more of your inputs are not greater than zero.\nUnable to continue. Program terminated.')  *elif* int(y) <= int(x):  print('You did not enter a value of y that is greater than x.\nUnable to continue. Program terminated.')  *else*:  *# Use a for loop to calculate the sum of numbers from x to y and store the final value in the variable sum\_of\_numbers*  x = int(x)  y = int(y)  sum\_of\_numbers = 0  *for* i *in* list(range(x,y+1)):  sum\_of\_numbers += i  *# display sum\_of\_numbers to the user*  print(f'The sum of numbers between {x} and {y} is {sum\_of\_numbers}') |

|  |
| --- |
| Text  Description automatically generated |

# Section 10 Tasks

### Task 2: Odd and Even

|  |
| --- |
| *import* random  *# define a function called oddandeven(numbers\_list)*  *# takes one list and returns two outputs of two lists*  *# one with even numbers, the other with odd numbers*  def oddandeven(*numbers\_list*):  even\_numbers = []  odd\_numbers = []  *for* i *in* *numbers\_list*:  *if* i % 2 == 0:  even\_numbers.append(i)  *else*:  odd\_numbers.append(i)  print(f'Odd: {odd\_numbers}\n')  print(f'Even: {even\_numbers}')  *# generate a list with 100 random numbers in the range of 1 to 1000 and store in original\_list*  original\_list = [random.randint(1,1000) *for* x *in* range(0,100)]  print(f'Original List: {original\_list}\n')  oddandeven(original\_list) |

|  |
| --- |
| Text  Description automatically generated |