

Python Programming for Data Science [BDS306B]

Laboratory Manual

1) Develop a python program to read n digit integer number, and separate the integer number and display each digit. [Hint: input:5678 output: 5 6 7 8, use: floor and mod operators)

Code:

Method-1

```
# Read an n-digit integer
n = int(input("Enter the number of digits: "))
if n > 0:
    number = int(input(f"Enter a {n}-digit integer: "))

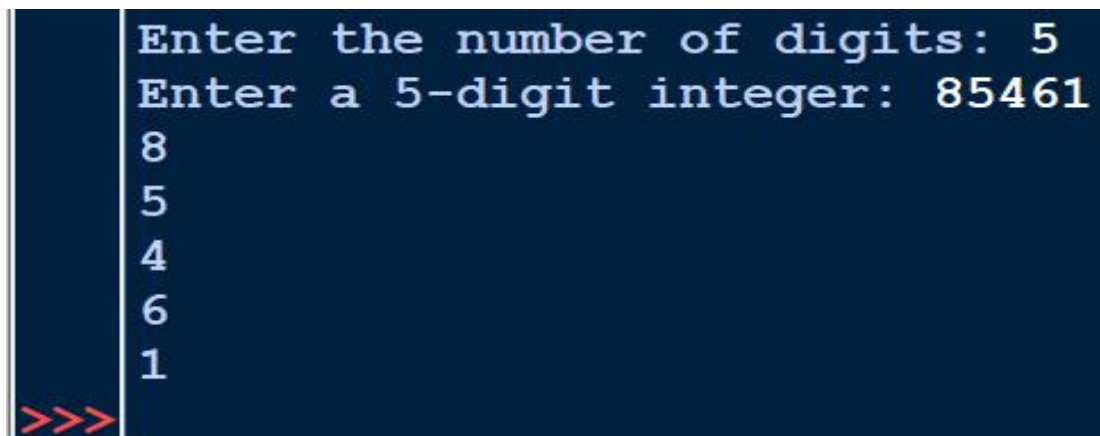
    # Check if the entered number has n digits
    if number >= 10 ** (n - 1) and number < 10 ** n:
        # Separate and display each digit using floor and mod operators
        for i in range(n):
            digit = number // (10 ** (n - 1))
            #print(digit, end=' ')
            print(digit)
            number %= 10 ** (n - 1)
            n -= 1
        else:
            print(f"Please enter a {n}-digit integer.")
    else:
        print("Please enter a valid number of digits (greater than 0).")
```

Method-2

```
# Read an n-digit integer
n = int(input("Enter the number of digits: "))
if n > 0:
    number = int(input(f"Enter a {n}-digit integer: "))

    # To check if the entered number has n digits
```

```
if len(str(number)) == n:
    # Separate and display each digit using floor and mod operators
    for i in range(n):
        digit = number // (10 ** (n - 1))
        print(digit)
        number %= 10 ** (n - 1)
        n -= 1
else:
    print(f'Please enter a {n}-digit integer.')
else:
    print("Please enter a valid number of digits (greater than 0).")
```

Output:

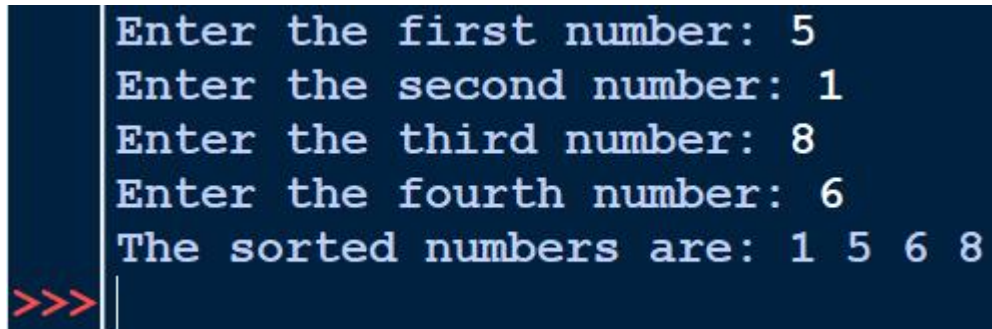
```
Enter the number of digits: 5
Enter a 5-digit integer: 85461
8
5
4
6
1
>>>
```

2) Develop a python program to accept 4 numbers and display them in sorted order using a minimum number of if else statements.

Code:

```
# Accept four numbers as input
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
c = int(input("Enter the third number: "))
d = int(input("Enter the fourth number: "))
# Sorting using conditional expressions
if a > b: a, b = b, a
if c > d: c, d = d, c
if a > c: a, c = c, a
```

```
if b > d: b, d = d, b
if b > c: b, c = c, b
# Display the sorted numbers
print("The sorted numbers are:", a, b, c, d)
```

Output:

```
Enter the first number: 5
Enter the second number: 1
Enter the third number: 8
Enter the fourth number: 6
The sorted numbers are: 1 5 6 8
>>>
```

3) Develop python scripts to Calculate the mean, median, mode, variance and standard deviation of n integer numbers.

Code:

```
# Accept n numbers from the user
n = int(input("Enter the number of elements: "))
numbers = [int(input(f'Enter number {i + 1}: ')) for i in range(n)]

# Calculate Mean
mean = sum(numbers) / n

# Calculate Median
numbers.sort()
if n % 2 == 0:
    median = (numbers[n // 2 - 1] + numbers[n // 2]) / 2
else:
    median = numbers[n // 2]

# Calculate Mode
frequency = {num: numbers.count(num) for num in numbers}
max_count = max(frequency.values())
```

```
mode = [k for k, v in frequency.items() if v == max_count]

if len(mode) == n: # No mode if all numbers appear the same number of times
    mode = "No mode"

# Calculate Variance (Sample Variance, divide by (n-1))
mean_difference_squared_sum = sum((x - mean) ** 2 for x in numbers)
variance_sample = mean_difference_squared_sum / (n - 1) if n > 1 else 0 # Avoid
division by 0
variance_population = mean_difference_squared_sum / n

# Calculate Standard Deviation
std_deviation_sample = variance_sample ** 0.5
std_deviation_population = variance_population ** 0.5

# Display results
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Mode: {mode}")
print(f"Sample Variance: {variance_sample}")
print(f"Population Variance: {variance_population}")
print(f"Sample Standard Deviation: {std_deviation_sample}")
print(f"Population Standard Deviation: {std_deviation_population}")
```

Output:

```
Enter the number of elements: 5
Enter number 1: 2
Enter number 2: 4
Enter number 3: 6
Enter number 4: 8
Enter number 5: 10
Mean: 6.0
Median: 6
Mode: No mode
Sample Variance: 10.0
Population Variance: 8.0
Sample Standard Deviation: 3.1622776601683795
Population Standard Deviation: 2.8284271247461903
>>>
```

4) Develop a program for checking if a given n digit number is palindrome or not.
[hint: input 1221 output: palindrome, use // and % operator with loop statement]

Code:

```
# Accept the number from the user
number = int(input("Enter an n-digit number: "))

# Make a copy of the original number to compare later
original_number = number
reversed_number = 0

# Reverse the number using // and % operators
while number > 0:
    digit = number % 10 # Get the last digit
    reversed_number = reversed_number * 10 + digit # Build the reversed number
    number //= 10 # Remove the last digit

# Check if the original number is the same as the reversed number
if original_number == reversed_number:
    print("Palindrome")
else:
    print("Not a palindrome")
```

Output:

```

Enter an n-digit number: 1221
Palindrome
>>>
>>>
==== RESTART: D:\#VAI\#VCET\#A
Enter an n-digit number: 5342
Not a palindrome
>>>

```

5) Develop a python script to display a multiplication table for given integer n.

Code:

```

# Accept the integer from the user
n = int(input("Enter an integer for the multiplication table: "))

# Display the multiplication table
print(f"Multiplication Table for {n}:")
for i in range(1, 11): # Loop to generate table from 1 to 10
    result = n * i
    print(f"{n} x {i} = {result}")

```

Output:

```

Enter an integer for the multiplication table: 8
Multiplication Table for 8:
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
>>>

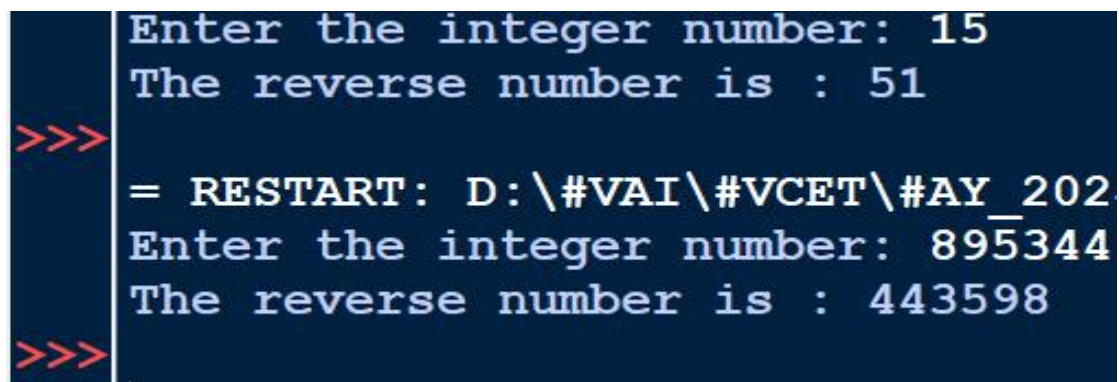
```

6) Develop a python script to rotate right about a given position in that list and display them. [hint: input [1,4,5,-10] position: 2, output: [-10,5,4,1]]

Code:

```
#Ask for enter the number from the use
number = int(input("Enter the integer number: "))
# Initiate value to null
revs_number = 0
#reverse the integer number using the while loop
while (number > 0):
    # Logic
    remainder = number % 10
    revs_number = (revs_number * 10) + remainder
    number = number // 10
    #Display the result
print("The reverse number is : {}".format(revs_number))
```

Output:



```
>>> Enter the integer number: 15
The reverse number is : 51
>>> = RESTART: D:\#VAI\#VCET\#AY_202
Enter the integer number: 895344
The reverse number is : 443598
>>>
```

7) Develop Write a python script to interchange the digits of a given integer number. [hint: input: 23456, interchange: 3 and 5 output: 25436]

Code:

```
def interchange_digits(number, pos1, pos2):
    # Convert the number to a string
    number_str = str(number)

    # Convert the string to a list of characters for easy manipulation
    number_list = list(number_str)
```



```
# Swap the digits at the given positions (1-based index)
number_list[pos1 - 1], number_list[pos2 - 1] = number_list[pos2 - 1],
number_list[pos1 - 1]

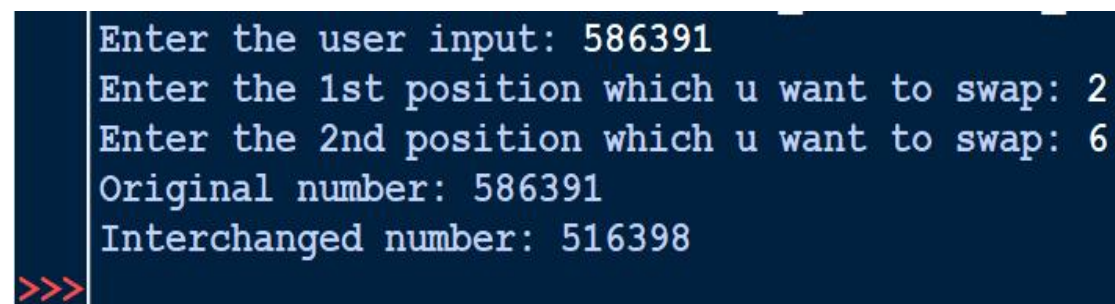
# Convert the list back to a string and then to an integer
interchanged_number = int("".join(number_list))

return interchanged_number

# Example usage
#number = 23456
number=int(input('Enter the user input: '))
pos1=int(input('Enter the 1st position which u want to swap: '))
pos2= int(input('Enter the 2nd position which u want to swap: '))

#pos1, pos2 = 3, 5 # Interchange the 3rd and 5th digits

interchanged_number = interchange_digits(number, pos1, pos2)
print("Original number:", number)
print("Interchanged number:", interchanged_number)
```

Output:

```
>>> Enter the user input: 586391
      Enter the 1st position which u want to swap: 2
      Enter the 2nd position which u want to swap: 6
      Original number: 586391
      Interchanged number: 516398
>>>
```

8) Develop a python program to capitalize a given list of strings. [hint: [hello, good, how, simple] output: [Hello, Good, How, Simple]

Code:

```
def capitalize_strings(string_list):
    # Capitalize each string in the list
    return [s.capitalize() for s in string_list]
```

```
"""The capitalize() function in Python is used to
capitalize the first letter of a string. """
# Example usage
input_strings = ['hello', 'good', 'how', 'simple']
capitalized_strings = capitalize_strings(input_strings)
print(f"Input_list: {input_strings}")
print(f"Capitalized_List: {capitalized_strings}")
```

Output:

```
Input_list: ['hello', 'good', 'how', 'simple']
Capitalized_List: ['Hello', 'Good', 'How', 'Simple']
```

9) Using a dictionary, Develop a python program to determine and print the number of duplicate words in a sentence.

Code:

```
def count_duplicate_words(sentence):
    # Split the sentence into words
    words = sentence.split()

    # Create a dictionary to store word counts
    word_count = {}

    # Count occurrences of each word
    for word in words:
        # Convert to lowercase for case-insensitivity
        word = word.lower()
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1

    # Find and print duplicate words
    duplicates = {word: count for word, count in word_count.items() if count > 1}
```

```

if duplicates:
    print("Duplicate words and their counts:")
    for word, count in duplicates.items():
        print(f"{word}: {count}")
else:
    print("No duplicate words found.")

```

Example usage

```

sentence = input("Enter a sentence: ")
count_duplicate_words(sentence)

```

Output:

```

Enter a sentence: "big black bug bit a big black dog on his big black nose"
Duplicate words and their counts:
black: 3
big: 2

```

10) Develop python program to read Numpy array and print row (sum,mean std) and column (sum,mean,std)

Code:

```

from numpy import asarray
# define data as a list
data = [[1,2,3], [4,5,6]]
# convert to a numpy array
data = asarray(data)
# summarize the array content
print("Numpy Array")
print(data)
print("_____")
# sum data by row & column
totalr = data.sum(axis=1)
totalc = data.sum(axis=0)
# summarize the result
print("Row Sum")
print(totalr)

```

```
print("Column Sum")
print(totalc)
print("_____")
meanr = data.mean(axis=1)
meanc = data.mean(axis=0)
print("Row Mean")
print(meanr)
print("Column Mean")
print(meanc)
print("_____")
stdr = data.std(axis=1)
stdc = data.std(axis=0)
print("Row STD")
print(stdr)
print("Column STD")
print(stdc)
```

Output:

```
Numpy Array
[[1 2 3]
 [4 5 6]]

Row Sum
[ 6 15]
Column Sum
[5 7 9]

Row Mean
[2. 5.]
Column Mean
[2.5 3.5 4.5]

Row STD
[0.81649658 0.81649658]
Column STD
[1.5 1.5 1.5]
```

11) Develop a python program to read and print in the console CSV file**Code:**

```
import csv

file = open('employee_birthday.csv', mode='r')
csv_reader = csv.reader(file, delimiter=',')
line_count = 0
for row in csv_reader:
    if line_count == 0:
        #print(row)
        #print(type(row))
        #break
        print(f'Column names are {", ".join(row)}')
        line_count += 1
    else:
        print(f'\t{row[0]} works in the {row[1]} department, and was born in {row[2]}.')
        line_count += 1

print(f'Processed {line_count} lines.')
file.close()
```

Output:

```
Column names are name , department, birthday_month
    John Smith works in the Accounting department, and was born in november.
    Erica Meyers works in the IT department, and was born in march.
    Rocky works in the HR department, and was born in april.
Processed 4 lines.
```

Note:

CSV file:

File name: employee_birthday.csv

	A	B	C
1	name	department	birthday_month
2	John Smith	Accounting	november
3	Erica Meyers	IT	march
4	Rocky	HR	april

12) Develop a python program to read a HTML file with basic tags, and construct a dictionary and display the same in the console.

Code:

```
import pandas as pd

def read_html_to_dict(html_file):
    # Read all tables from the HTML file
    tables = pd.read_html(html_file)

    # Initialize an empty dictionary to store each table
    tables_dict = {}

    # Loop over each table and convert it to a dictionary
    for i, table in enumerate(tables):
        # Convert the DataFrame to a dictionary (each table
        # will be its own dictionary entry)
        table_dict = table.to_dict(orient='list')
        tables_dict[f'Table_{i+1}'] = table_dict
    return tables_dict

if __name__ == '__main__':
    html_file = 'sample.html' # Replace with your HTML file path
    result = read_html_to_dict(html_file)
    # Display the dictionary
    print(result)
```

HTML code:

```
1 <html>
2   <body>
3     <table>
4       <tr><th>Name</th><th>Age</th></tr>
5       <tr><td>Alice</td><td>24</td></tr>
6       <tr><td>Bob</td><td>27</td></tr>
7     </table>
8     <table>
9       <tr><th>Country</th><th>Capital</th></tr>
10      <tr><td>France</td><td>Paris</td></tr>
11      <tr><td>Italy</td><td>Rome</td></tr>
12    </table>
13  </body>
14 </html>
```

Name Age

Alice 24

Bob 27

Country Capital

France Paris

Italy Rome

Note: Type the html scripts in the note pad or any other text file and then, save it as [filename.html](#) format. Keep the html file in the same folder where your python code is saved.

Output:

```
{'Table_1': {'Name': ['Alice', 'Bob'], 'Age': [24, 27]}, | 'Table_2': {'Country': ['France',  
'Italy'], 'Capital': ['Paris', 'Rome']}}
```