Ex. No: 1	NUMERIC DATATYPE
22 / 02 / 22	

AIM:

To learn about numeric datatypes in Python.

QUESTION 1:

Read an integer from the user and check whether the number is a palindrome.

ALGORITHM:

Step 1: Read input integer as a string.
Step 2: Append characters starting from the end using negative indexing to the reverse string.
Step 3: Compare using '=' operator.
Step 4: Display output.

QUESTION 2:

Read an integer from the user and print its binary equivalent.

ALGORITHM:

Step 1: Read input integer.
Step 2: Get binary equivalent using 'bin( )' function.
Step 3: Exclude '0b' from output and print output.

QUESTION 3:

Read an integer from the user and print its prime factors.

ALGORITHM:

Step 1: Read input integer 'num'.
Step 2: If 'num' is '1',
Print that it has no prime factors.
Else,
While 'num' is not '1',
For every number 'div' from '2' to 'num' (inclusive),
If 'num' is divisible by 'div',
Print 'div'.
Divide 'num' by 'div'.

Program 1:

```
"""
Read an integer from the user and check whether the number is a palindrome.
"""

# Input
orig = input("Enter an integer: ")
rev = ''

# Reverse the integer
for ind in range(1, len(orig) + 1):
    rev += orig[-ind]    # Negative index

# Display reversed number
print("\nOriginal: ", orig)
print("Reverse : ", rev)

# Check if palindrome or not + Output
if (orig == rev):
    print("\nIt is plaindrome.")
else:
    print("\nIt is not palindrome.")

'''
(or)

# Reverse using string slicing
rev = orig[::-1].lower()

String Slicing:
    "<string>"[<start_index>:<end_index>:<step>]

    Default <step> = 1
    Default <start_index> = 0
    Default <end_index> = -1 [-ve Index]

E.g.
    "Hello World"[1:8] -> "ello Wo"
    "Hello World"[::2] -> "HloWrd"
'''
```

INPUT / OUTPUT:

Enter an integer: 1092

Original: 1092
Reverse: 2901

It is not palindrome.

Enter an integer: 96169

Original: 96169
Reverse: 96169

It is palindrome.

Program 2:

```
"""
Read an integer from the user and print its binary equivalent.
"""

num = int(input("Enter an integer: "))
binary = bin(num)    # bin(<num>) -> Built-in function

print("Binary: ", end = "")

for i in range(2, len(binary)):
    print(binary[i], end = "")

'''
(or)

# Input
num = int(input("Enter an integer: "))

# String slicing
binary = bin(num)[2:]

# Output
print("Binary:", binary)
'''
```

INPUT / OUTPUT:

Enter an integer: 35
Binary: 100011

Enter an integer: 23987
Binary: 101110110110011

Program 3:

```
"""
Read an integer from the user and print its prime factors.
"""

# Input
num = int(input("Enter an integer: "))

if (num == 1):
    print("'1' has no prime factors.'")
else:
    print("Prime factor(s): ", end = '')

    while (num != 1):
        for div in range(2, num + 1):
            if (num % div == 0):
                # Print last prime factor
                if (int(num/div) == 1):
                    print(div, end = '.')
                    num = int(num / div)
                    break
                # Print prime factors
                else:
                    print(div, end = ', ')
                    num = int(num / div)
                    break
```

RESULT:

Concepts related numeric datatypes were studied.

INPUT / OUTPUT:

Enter an integer: 1092
Prime factor(s): 2, 2, 3, 7, 13.

Enter an integer: 100
Prime factor(s): 2, 2, 5, 5.

Enter an integer: 314159265
Prime factor(s): 3, 3, 5, 7, 127, 7853.

Ex. No: 2	STRING DATATYPE
03 / 03 / 22	

AIM:

To learn about string datatype in Python.

QUESTION:

Read a string from the user, assuming it is a password creation process. When the user sets a password, the following conditions should be satisfied:

a) At least one lowercase alphabet is required.
b) At least one uppercase alphabet is required.
c) At least one digit is required.
d) At least one special character is required.
e) The length of the string should be at least 8 and maximum 16.

1. If the password entered does not satisfy the above conditions, then throw out the appropriate message.
2. If the password entered contains four occurrences of conditions a) to d), then print that the password strength is strong.
3. If the password entered contains two or three occurrences of conditions a) to d), then print that the password strength is medium.
4. If the password entered contains only one occurrence of d), then print that the password strength is poor.

ALGORITHM:

Step 1: Until the user enters a valid password,
    Read input string ‘pwd’ from user.
    If length of ‘pwd’ is not between ‘8’ and ‘16’ characters,
        Redo Step 1.
    If ‘pwd’ has no uppercase, lowercase, special characters, and digits,
        Throw appropriate error message.
        Redo Step 1.
Step 2: Find count of uppercase, lowercase, and special characters, and digits in ‘pwd’.
Step 3: If all counts are ‘4’ or greater,
    Print that the password strength is strong.
    If all counts are ‘2’ or ‘3’,
    Print that the password strength is medium.
    If all counts are ‘1’,
    Print that the password strength is weak.

Program:

```
"""
Read a string from the user, assuming it is a password creation process. When
the user sets a password, the following conditions should be satisfied:

a) At least one lowercase alphabet is required.
b) At least one uppercase alphabet is required.
c) At least one digit is required.
d) At least one special character is required.
e) The length of the string should be at least 8 and maximum 16.


If the password entered does not satisfy the above conditions, then throw
out the appropriate message.

If the password entered contains four occurrences of conditions a) to d),
then print that the password strength is strong.

If the password entered contains two or three occurrences of conditions a)
to d), then print that the password strength is medium.

If the password entered contains only one occurrence of a) to d), then print
that the password strength is poor.
"""

# Check for invalid passwords
while True:
    pwd = input("Enter password: ")

    if (pwd.isupper()):
        print("At least 1 lowercase alphabet is required.")
        continue
    if (pwd.islower()):
        print("At least 1 uppercase alphabet is required.")
        continue
    if (pwd.isalpha()):
        print("At least 1 digit is required.")
        continue
    if (pwd.isnumeric()):
        print("At least 1 alphabet is required.")
        continue
    if (pwd.isalnum()):
        print("At least 1 special character is required.")
        continue
    if (len(pwd) < 8 or len(pwd) > 16):
        print("Password must be 8 - 16 characters long.")
        continue
    break

print("Password accepted.")
```

```
# Calculate password strength
lowerCount = 0
upperCount = 0
digitCount = 0
specialCount = 0

for i in range(len(pwd)):
    if pwd[i].islower():
        lowerCount += 1
    elif pwd[i].isupper():
        upperCount += 1
    elif pwd[i].isdigit():
        digitCount += 1
    else:
        specialCount += 1;

if (lowerCount > 3 and upperCount > 3 and digitCount > 3 and specialCount >
3):
    print("\nPassword strength: Strong.")
elif (lowerCount > 1 and upperCount > 1 and digitCount > 1 and specialCount
> 1):
    print("\nPassword strength: Medium.")
else:
    print("\nPassword strength: Weak.")
```

RESULT:

Concepts related string datatype were studied.

INPUT / OUTPUT:

Enter password: home
At least 1 uppercase alphabet is required.

Enter password: Home
At least 1 digit is required.

Enter password: Home12
At least 1 special character is required.

Enter password: Home@12
Password must be 8 - 16 characters long.

Enter password: HomeHome@12
Password accepted.

Password strength: Weak.

Enter password: Home@Home. 12
Password accepted.

Password strength: Medium.

Enter password: HOME@home.123#4!
Password accepted.

Password strength: Strong.

Ex. No: 3	STRING MANIPULATIONS - I
08 / 03 / 22	

AIM:

To learn string manipulations in Python.

QUESTION 1:

Read a string and check if the first character is occurring more than once in the string. If yes, replace the occurrence by '$' from the second instance.

E.g.: input: restart, output : resta$t

ALGORITHM:

Step 1: Read string input 'str1' from user.
Step 2: Append the first character of 'str1' to 'str2'.
Step 3: Replace all occurrences of first character of 'str1' with '$'.
Step 4: Append remaining characters of 'str1' to 'str2'.
Step 5: Print output.

QUESTION 2:

Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

ALGORITHM:

Step 1: Read string input 'str1' from user.
Step 2: If length of 'str1' is greater than or equal to '3',
    If 'str1' ends with 'ing',
        Append 'ly' to the string
    Else,
        Append 'ing' to the string

Step 3: Print string 'str1' as output.

Program 1:

```
"""
Read a string and check if the first character is occurring
more than once in the string. If yes, replace the occurrence
by ‘$’ from the second instance.

E.g.: input: restart, output : resta$t
"""

# Input
str1 = input("Enter a string: ")

# Replace recurring first character by '$' from second instance onwards
str2 = str1[0] + str1[1:].replace(str1[0], '$')

# Output
print("Output:", str2)
```

INPUT / OUTPUT:

Enter a string: restart
Output: resta$t

Enter a string: arrays are faster than lists
Output: arr$ys $re f$ster th$n lists

Program 2:

```
"""
Write a Python program to add 'ing' at the end of a given
string (length should be at least 3). If the given string
already ends with 'ing' then add 'ly' instead. If the string
length of the given string is less than 3, leave it unchanged.
"""

# Input
str1 = input("Enter a string: ")

# Add 'ing' or 'ly' (if string ends with 'ing' already) if string length >
3
if len(str1) >= 3:
    if (str1.endswith("ing")):
        str1 += "ly"
    else:
        str1 += "ing"

# Output
print("Output string:", str1)
```

RESULT:

The concepts related to string manipulations were studied.

INPUT / OUTPUT:

Enter a string: interest
Output string: interesting

Enter a string: amazing
Output string: amazingly

Enter a string: hi
Output string: hi

Ex. No: 4 22 / 03 / 22	STRING MANIPULATIONS - II

AIM:

To learn more string manipulations in Python.

QUESTION 1:

Write a Python function that takes two strings and return a string where the first two characters of each string are swapped and separated by space. E.g.: 'abc', 'xyz' o/p: 'xyc abz'. Validate the input such that if the length of either of the strings is less than 3, print error message saying "Enter a valid string with length more than 3".

ALGORITHM:

Step 1: Read string inputs 'str1' & 'str2' from user.
Step 2: If either or both string lengths are less than 3,
    Print error message.
    End program.
  Else,
    Append first '2' characters of 'str2', 2nd character till last character of 'str1', whitespace, first '2' characters of 'str1', and 2nd character till last character of 'str2' in the same order to the output string.
Step 3: Print output string.

QUESTION 2:

Write a Python function that takes a string and an integer 'n'. The function should return a string after removing the 'n'th character in the string.

E.g. i/p: 'Python', 3

o/p: 'Pyhon'

ALGORITHM:

Step 1: Read string input 'str1' & integer 'n' from user.
Step 2: If 'n' is not negative or zero,
    Append front part of string until 'nth' character to output string.
    Append remaining part of string excluding 'nth' character to output string.
Step 3: Print output string.

QUESTION 3:

Write a Python function to get a string made of 3 copies of the last 4 characters of a specified string (length must be at least 5)

i/p: python o/p: ‘thonthonthon’

ALGORITHM:

Step 1: Read string input ‘string’.
Step 2: If length of ‘string’ is less than 5,
        Print error message.
    Else,
        Append last 4 characters of ‘string’ x3 times into output string
Step 3: Print the output string.

INPUT / OUTPUT:

Enter a string: abc
Enter another string: xyz
Output: xyc abz

Enter a string: ab
Enter another string: xyz
Error: Enter a valid string with length more than 3.

Program 1:

```
"""
Write a Python function that takes two strings and return a string
where the first two characters of each string are swapped and separated
by space.
Eg: 'abc', 'xyz' o/p: 'xyc abz'.

Validate the input such that if the length of either of the strings
is less than 3, print error message saying “Enter a valid string with
length more than 3”
"""

def swapandsep(str1, str2):
    '''
    Function that returns a concatenated string with the first two
characters of ‘str1’ and ‘str2’ swapped and separated by space.

    Function takes two arguments:
    arg1: str
    arg2: str
    return: str
    '''
    # Input validation
    if len(str1) < 3 or len(str2) < 3:
        return
    else:
        str3 = str2[:2] + str1[2:] + ' ' + str1[:2] + str2[2:]
        return str3


# Input
str1 = input("Enter a string: ")
str2 = input("Enter another string: ")

# Function call
str3 = swapandsep(str1, str2)

# Output
if str3 == None:
    print("\nError: Enter a valid string with length more than 3.")
else:
    print("\nOutput:", str3)

'''
Note: 'return' = 'return None'
'''
```

Program 2:

```
"""
Write a python function that takes a string and an integer n.
The function should return a string after removing the ‘n’th
character in the string.

E.g.
i/p: ‘Python’, 3
o/p: ‘Pyhon’
"""

def removechar(string, n):
    '''
    Function that returns string with 'n'th character in string removed.

    Function takes two arguments:
    arg1: str
    arg2: int
    return: str
    '''
    if n > 0:
        return string[:n-1] + string[n:]
    # Function will return 'None' if 'return' is not specified

# Input
string = input("Enter a string: ")
n = int(input("Enter an integer: "))

# Function call
string = removechar(string, n)

# Output
if string == None:
    print("Invalid integer.")
else:
    print(string)
```

INPUT / OUTPUT:

Enter a string: Python
Enter an integer: 3
Pyhon

Enter a string: Shiv Nadar University
Enter an integer: 0
Invalid integer.

Enter a string: Python Lab
Enter an integer: 7
PythonLab

Program 3:

```
"""
Write a Python function to get a string made of 3 copies of the last 4
characters of a specified string (length must be at least 5).

i/p: python
o/p: ‘thonthonthon’
"""

def repeatLastFour(string):
    '''
    Function that returns string with 3 copies of the last
    4 characters of the argument string.

    Function takes one argument:
    arg1: str
    return: str
    '''
    if (len(string) < 5):
        return None
    else:
        return string[-4:] * 3

# Input
string = input("Enter a string: ")

# Function call
newString = repeatLastFour(string)

# Output
if newString == None:
    print("Invalid string.")
else:
    print(newString)
```

RESULT:

More concepts related to string manipulations were studied.

INPUT / OUTPUT:

Enter a string: python
thonthonthon

Enter a string: KaPoww
PowwPowwPoww

Enter a string: Hi
Invalid string.

Ex. No: 5	SLICING OPERATIONS
29 / 03 / 22	

AIM:

To learn about slicing operations in Python.

QUESTION 1:

Using the slicing operation on string representing URLs of websites, check the following and print appropriate messages:

1. Whether the string starts with any of "http:", "https:" or "ftp:". If yes, then print the message that it starts with the appropriate sub-string.
2. Whether the string ends with any of ".com" , ".org" or ".in". If yes, then print the message that it ends with the appropriate sub-string.
3. If none of these conditions satisfy, then print appropriate message that the string doesn't start or end with any of the above given sub-strings.

Note:

1. The string should be accepted using input( ) function and the strings entered are URLs of various websites.
2. The sub-strings at the beginning and end should be checked using both positive and negative index slicing inside the conditional checks.
3. Built-in or user defined functions should not be used in this code.

ALGORITHM:

Step 1: Read input string from user.
Step 2: If string starts with "http:", "https:", or "ftp:",
        Print that it starts with the respective sub-string.
        Else,
        Set start string condition to false.
Step 3: If string ends with ".com" , ".org", or ".in",
        Print that it ends with the respective sub-string.
        Else,
        Set end string condition to false.
Step 4: If start and end string conditions are false,
        Print that the string does not start or end with given sub-strings.

QUESTION 2:

For the previous problem, write two functions s_startsWith( ) and s_endsWith( ) which takes this string as argument to check whether it starts with any of "http:", "https:" or "ftp:" or ends with any of ".com" , ".org" or ".in". Print appropriate messages as above.

Note:

1. The string should be accepted using input( ) function and the strings entered are URLs of various websites.
2. The sub-strings at the beginning and end should be checked using both positive and negative index slicing inside the conditional checks which will be used within the function definition.
3. Use only user defined functions in this code. Do not use any built-in function.

ALGORITHM:

Step 1: Read input string from user.
Step 2: If string starts with "http:", "https:", or "ftp:" (use string slicing),
        Print that it starts with the respective sub-string.
    Else,
        Print that it does not start with the respective sub-string.
Step 3: If string ends with ".com" , ".org", or ".in",
        Print that it ends with the respective sub-string.
    Else,
        Print that it does not end with the respective sub-string.

Program 1:

```
"""
Using the slicing operation on string representing URLs of
websites, check the following and print appropriate messages:

    Whether the string starts with any of "http:", "https:" or
    "ftp:". If yes, then print the message that it starts
    with the appropriate sub-string.

    Whether the string ends with any of ".com" , ".org" or ".in".
    If yes, then print the message that it ends with the appropriate
    sub-string.

    If none of these conditions satisfy, then print appropriate
    message that the string doesn't start or end with any of the
    above given sub-strings.

Note:
    The string should be accepted using input(  ) function and
    the strings entered are URLs of various websites.
    The sub-strings at the beginning and end should be checked
    using both positive and negative index slicing inside the
    conditional checks.

    Built-in or user defined functions should not be used in
    this code.
"""

# Input
string = input("Enter a string: ")

startCond = endCond = False

# Check start - positive index slicing
if (string[0:4] == "ftp:"):
    print("\nIt starts with 'ftp:'.")
    startCond = True
elif (string[0:5] == "http:"):
    print("\nIt starts with 'http:'.")
    startCond = True
elif (string[0:6] == "https:"):
    print("\nIt starts with 'https:'.")
    startCond = True

# Check end - negative index slicing
if (string[-3:] == ".in"):
    print("It ends with '.in'.")
    endCond = True
elif (string[-4:] == ".com"):
    print("It ends with '.com'.")
    endCond = True
```

```
elif (string[-4:] == ".org"):
    print("It ends with '.org'.")
    endCond = True

if (startCond == False and endCond == False):
    print("\nThe string doesn't start or end with any of the above given
sub-strings. \n")
```

INPUT / OUTPUT:

Enter a string: https://www.snuchennai.edu.in
It starts with 'https:'.
It ends with '.in'.

Enter a string: www.shuchennai.com
It ends with '.com'.

Enter a string: Hello, World!
The string doesn't start or end with any of the above given sub-strings.

Program 2:

```
"""
For the above problem, write two functions s_startsWith(  )
and s_endsWith(  ) which takes this string as argument to check whether it
starts with any of "http:", "https:" or "ftp:" or ends with
any of ".com" , ".org" or ".in". Print appropriate messages as above.

Note:
    The string should be accepted using input(  ) function and the strings
    entered are URLs of various websites.

    The sub-strings at the beginning and end should be checked using both
    positive and negative index slicing inside the conditional checks which
    will be used within the function definition.

    Use only user defined functions in this code. Do not use any built-in
    function.
"""

def s_startsWith(string):
    '''
    Function checks whether the argument string
    starts with any of "http:", "https:" or "ftp:".

    Function takes one argument:
    arg1: str
    return: str (or) None
    '''
    # Check start
    if (string[0:4] == "ftp:"):
        return '"fttp:"'
    elif (string[0:5] == "http:"):
        return '""http:"'
    elif (string[0:6] == "https:"):
        return '"https:"'
    else:
        return None


def s_endsWith(string):
    '''
    Function checks whether the argument string
    ends with any of ".com" , ".org" or ".in".

    Function takes one argument:
    arg1: str
    return: str (or) None
    '''
    # Check end
    if (string[-3:] == ".in"):
        return '".in"'
```

```
    elif (string[-4:] == ".com"):
        return '".com"'
    elif (string[-4:] == ".org"):
        return '".org"'
    else:
        return None

# Input
string = input("Enter a string: ")

# Function call + Output
if s_startsWith(string) == None:
    print("\nThe string doesn't start with \"http:\", \"https:\" or
\"ftp:\". \n")
else:
    print("\nIt starts with %s." % s_startsWith(string))

if s_endsWith(string) == None:
    print("\nThe string doesn't end with \".com\" , \".org\" or \".in\".
\n")
else:
    print("\nIt ends with %s." % s_endsWith(string))
```

RESULT:

The concepts related to slicing operations were studied.

INPUT / OUTPUT:

Enter a string: https://www.lms.snuchennai.edu.in

It starts with "https:".

It ends with ".in".

Enter a string: www.google.com

The string doesn't start with "http:", "https:" or "ftp:".

It ends with ".com".

Enter a string: Shiv Nadar University

The string doesn't start with "http:", "https:" or "ftp:".

The string doesn't end with ".com', ".org" or ".in".

Ex. No: 6	FUNCTION HANDLING
05 / 04 / 22	

AIM:

To learn function handling in Python.

QUESTION 1:

Write a Python function that takes two strings and check if one string is the anagram of the second string or not.

ALGORITHM:

Step 1: Read input strings ‘str1’ and ‘str2’ from user.
Step 2: Check if each character of ‘str1’ is in ‘str2’.
Step 3: Check the number of times all characters occur in ‘str1’, and ‘str2’. If not equal,
Print that one string is not the anagram of the second string.
Else,
Print that one string is the anagram of the second string.

QUESTION 2:

Write a Python script to build ATM Machine functionality.

Acceptance Criteria:

a) Create a Python function which asks ATM PIN as input and if the PIN is valid it should display Withdraw and Check Account Balance option. (Use string functions to check if a PIN has only 4 numeric characters).
b) Create a Python function which allows user to select the option.
c) Create a separate Python function which implements check account balance and withdraw functionality.
d) After the successful withdrawal account balance should reduce.
e) In an user enters invalid withdrawal, you should notify the user with error message .

ALGORITHM:

Step 1: Set initial account balance = 25000.
Step 2: Input PIN as string.
Step 3: If PIN is not 4 characters long,
Print “Invalid ATM PIN”.
Exit program.
Step 4: If PIN has any character other than digits,

Print “Invalid ATM PIN”.
Exit program.

Step 5: Print options menu.

Step 6: If option is to exit,
Exit program.
If option is to check balance,
Print available balance.
If option is to withdraw,
Input amount to withdraw.
If amount is invalid (‘0’ or negative),
Print invalid amount entered.
If amount > available balance,
Print insufficient funds available.
Else:
Deduct amount from available balance.

Program 1:

```
"""
Write a python function that takes two strings and check if one
string is the anagram of the second string or not.
"""

def isanagram(str1, str2):
    '''
    Takes two strings and returns 'True' if 'str1' is the anagram
    of the 'str2' and 'False' if it is not.

    Function takes one argument:
    arg1: str
    arg2: str

    return: bool
    '''
    # Convert all characters to lowercase
    str1 = str1.lower()
    str2 = str2.lower()

    # Check if all characters in 'str1' are in 'str2'
    for char in str1:
        if char not in str2:
            return False
        else:
            pass

    # Check number of times the character occurs in both strings
    for char in str1:
        if str1.count(char) != str2.count(char):
            return False
        else:
            pass

    return True

# Input
str1 = input("Enter a string: ")
str2 = input("Enter another string: ")

# Function call + Output
if isanagram(str1, str2):
    print()
    print(str1, " is the anagram of ", str2, ".", sep="")
else:
    print()
    print(str1, " is not the anagram of ", str2, ".", sep="")

'''
```

```
Note:
    Anagram is a word, phrase, or name formed by
    rearranging the letters of another.
    E.g. 'spar', formed from 'rasp'.
'''
```

INPUT / OUTPUT:

Enter a string: ShivNadarUniversity
Enter another string: SityUnivShivNadar
ShivNadarUniversity is not the anagram of SityUnivShivNadar,

Enter a string: Astronomer
Enter another string: MoonStarer
Astronomer is the anagram of MoonStarer.

Program 2:

```
"""
Write a Python script to build ATM Machine functionality .

Acceptance Criteria:

a) Create a Python function which asks ATM PIN as input and if the
PIN is valid it should display Withdraw and Check Account Balance option.
(Use string functions to check if a PIN has only 4 numeric characters).

b) Create a Python function which allows user to select the option.

c) Create a separate Python function which implements check account
balance and withdraw functionality.

d) After the successful withdrawal account balance should reduce.

e) If a user enters invalid withdrawal, you should notify the user
with error message.
"""

import os
import time

def withdraw(bal):
    '''
    Reduces the account balance if the withdrawal is successful. Prints
    error message otherwise.

    Function takes 1 argument:
    arg1: int

    return: None
    '''
    # Input
    amt = int(input("Enter amount to withdraw: "))

    if amt < 0:
        print("Invalid amount entered.")
    elif amt > bal:
        print("Invalid withdrawal. Sufficient funds not available.")
    else:
        bal -= amt

    return bal

def checkAccountBalance(bal):
    '''
    Prints the user's account balance.
```

```
    Function takes 1 argument:
    arg1: int

    return: None
    '''
    # Output
    print("\nAvailable Balance: INR", float(bal))

    return

def selectOption(bal):
    '''
    Allows the user to select an option.

    Function takes 1 argument:
    arg1: int

    return: None
    '''
    while True:
        # Input
        option = int(input("Enter an option: "))
        if option == 0:
            break
        elif option == 1:
            bal = withdraw(bal)
        elif option == 2:
            checkAccountBalance(bal)
        else:
            print("Invalid option.")

    print("\nProgram ended.")
    return


def isvalidpin():
    '''
    Takes ATM PIN as input and if the PIN is valid displays
    Withdraw and Check Account Balance option.

    Function takes no arguments.

    return: None
    '''

    # Input
    pin = input("Enter the ATM PIN: ")

    '''
    # Clear terminal screen
    os.system('clear')
```

```
    # Pause program for 0.5 seconds
    time.sleep(0.5)
    '''

    # Check if PIN is 4 characters long
    if len(pin) > 4 or len(pin) < 4:
        print("Invalid ATM PIN.")
        return

    # Check if PIN consists only of numbers
    for char in pin:
        if char.isdigit() != True:
            print("Invalid ATM PIN.")
            return

    # PIN is 4 characters long & contains only numbers
    print("\n*** Options Menu ***")
    print("1. Withdraw.")
    print("2. Check Account Balance.")
    print("0. Exit.")

    # Select an option
    selectOption(bal)

    return

# Initial account balance
bal = 25000

# Clear terminal screen
os.system('clear')

# Function call
isvalidpin()
```

RESULT:
The concepts related to function handling were studied.

INPUT / OUTPUT:

Enter the ATM PIN: 1001

*** Options Menu ***
1. Withdraw.
2. Check Account Balance.
0. Exit.

Enter an option: 2

Available Balance: IN 25000.0

Enter an option: 1

Enter amount to withdraw: 10000

Enter an option: 2

Available Balance: IN 15000.0

Enter an option: 1

Enter amount to withdraw: 25000
Invalid withdrawal. Sufficient funds not available.

Enter an option: 2

Available Balance: IN 15000.0

Enter an option: 0
Program ended.

Ex. No: 7	LIST CONTAINER
09/ 04 / 22	

AIM:

To learn about list containers in Python.

QUESTION 1:

Write a Python function "listdiff(l1,l2)" that takes two lists l1 and l2 as input and prints all values that are either in l1 or in l2 but not in both lists. You may assume that each input list is sorted in ascending order and has distinct values (i.e., no duplicates). You may also assume that the two lists have elements of the same type. Your output should also be in ascending order. Do *not* sort the output --- you should generate the output values in ascending order.

Here are some examples to show how your function should behave:

```
>>> listdiff([1,3,5],[1,2,4,6])
[2,3,4,5,6]
>>> listdiff([2,5],[2,5])
[ ]
```

ALGORITHM:

Step 1: Input 2 lists ‘l1’ and ‘l2’.
Step 2: Remove duplicate numbers in both lists.
Step 3: Sort both lists in ascending order.
Step 4: Declare a list ‘l3’.
Step 5: If both lists are the same,
    Return ‘[ ]’ (empty list).
  Else,
    Compare first elements of both lists.
    If elements are equal,
      Move to next element in both lists.
    Else,
      If element in ‘l1’ is smaller than element in ‘l2’,
        Append element in ‘l1’ to output list ‘l3’.
        Move to next element in ‘l1’.
      Else,
        Append element in ‘l2’ to output list ‘l3’.
        Move to next element in ‘l2’.
Step 6: Print output list ‘l3’.

QUESTION 2:

Write a function "minout(l)" that takes a list ‘l’ of distinct natural numbers (i.e. integers from the set {0,1,2,...}) and returns the smallest natural number not present in ‘l’. In other words, if 0 is not in ‘l’, then "minout(l)" returns 0, else if 0 is in ‘l’ but 1 is not in ‘l’, then "minout(l)" returns 1, etc.

Here are some examples to show how your function should behave:

```
>>> minout([1,3,2,4,17])
0
>>> minout([1,3,0,2,4])
5
```

ALGORITHM:

Step 1: Read input list ‘l’ from user.
Step 2: Remove duplicates from list.
Step 3: If ‘0’ is not in ‘l’,
        Return ‘0’.
    Else,
        num = 1
        If num in ‘l’,
            Increment ‘num’ by ‘1’.
        Else (‘num’ not in ‘l’),
            Print ‘num’.

Program 1:

```
"""
Write a Python function "listdiff(l1,l2)" that takes two
lists l1 and l2 as input and returns all values that are
either in l1 or in l2 but not in both lists.

You may assume that each input list is sorted in ascending
order and has distinct values (i.e., no duplicates).  You may
also assume that the two lists have elements of the same type.

Your output should also be in ascending order.  Do *not* sort
the output --- you should generate the output values in
ascending order.

Here are some examples to show how your function should behave:

>>> listdiff([1,3,5],[1,2,4,6])
[2,3,4,5,6]
>>> listdiff([2,5],[2,5])
[]
>>> listdiff(list(range(0,7,2)),list(range(1,8,2)))
```

```
[0,1,2,3,4,5,6,7]
"""

def listdiff(l1, l2):
    '''
    Takes two lists l1 and l2 as input and returns all values that are
    either in l1 or in l2 but not in both lists.

    Function takes two arguments:
    arg1: list
    arg2: list

    return: list
    '''
    # 'l2' must be smaller list
    if len(l1) < len(l2):
        temp = l1
        l1 = l2
        l2 = temp

    # Declare an empty list 'l3'
    l3 = []

    # Initialise 'i' & 'j'
    i, j = 0, 0

    if (l1 == l2):
        # Both lists are same
        return []
    else:
        # Different lists

        # Compare elements of both lists
        while (i < len(l1)):
            while (j < len(l1)):
                if j == len(l2):
                    # All elements in 'l2' were compared,
                    # Add remaining elements of 'l1' to 'l3
                    l3.extend(l1[i:])
                    return l3
                elif i == len(l1):
                    # All elements in 'l1' were compared,
                    # Add remaining elements of 'l2' to 'l3'
                    l3.extend(l2[j:])
                    return l3
                elif (l1[i] > l2[j]):
                    # Element in 'l2' is smaller than element in 'l1'
                    # Append smaller element to 'l3'
                    # Move to next element in 'l2'
                    l3.append(l2[j])
                    j += 1
                elif (l1[i] < l2[j]):
```

```
                    # Element in 'l1' is smaller than element in 'l2'
                    # Append smaller element to 'l3'
                    # Move to next element in 'l1'
                    l3.append(l1[i])
                    i += 1
                else:
                    # Elements are equal
                    # Move to next element in 'l1' & 'l2'
                    i += 1
                    j += 1

# Declare empty lists
l1, l2 = [], []

# Input
l1Len = int(input("Enter length of first list: "))

for i in range(l1Len):
    l1.append(int(input("Enter element: ")))

l2Len = int(input("Enter length of second list: "))

for i in range(l2Len):
    l2.append(int(input("Enter element: ")))

# Remove duplicates
l1 = list(set(l1))
l2 = list(set(l2))

# Sort list in ascending order
l1.sort()
l2.sort()

# Output
print("\nList 1:", l1)
print("List 2:", l2)
print("Note: Duplicates were removed and list was sorted in ascending
order.")

l3 = listdiff(l1, l2)    # Function call
print("\nOutput:", l3)
```

INPUT / OUTPUT:

Enter length of first list: 3

Enter element: 1

Enter element: 3

Enter element: 5

Enter length of second list: 4

Enter element: 1

Enter element: 2

Enter element: 4

Enter element: 6

List 1: [1, 3, 5]
List 2: [1, 2, 4, 6]
Note: Duplicates were removed and list was sorted in ascending order.

Output: [2, 3, 4, 5, 6]

Enter length of first list: 2

Enter element: 2

Enter element: 5

Enter length of second list: 2

Enter element: 2

Enter element: 5

List 1: [2, 5]
List 2: [2, 5]
Note: Duplicates were removed and list was sorted in ascending order.

Output: []

Enter length of first list: 4

Enter element: 0

Enter element: 2

Enter element: 4

Enter element: 6

Enter length of second list: 4

Enter element: 1

Enter element: 3

Enter element: 5

Enter element: 7

List 1: [0, 2, 4, 6]
List 2: [1, 3, 5, 71]
Note: Duplicates were removed and list was sorted in ascending order.

Output: [0, 1, 2, 3, 4, 5, 6, 7]

Program 2:

```
"""
Write a function "minout(l)" that takes a list 'l' of distinct
natural numbers (i.e. integers from the set {0,1,2,...}) and
returns the smallest natural number not present in 'l'.

In other words,
    if '0' is not in 'l', then "minout(l)" returns 0,
    else if '0' is in 'l' but '1' is not in 'l', them "minout(l)" returns 1,
etc.

Note that 'l' is NOT assumed to be sorted.

Here are some examples to show how your function should behave:

>>> minout([1,3,2,4,17])
0
>>> minout([1,3,0,2,4])
5
"""

def minout(l):
    '''
    Returns the smallest natural number which is not in argument list.
    Returns '0' if '0' is not in argument list

    Parameters
    ----------
    l : list
        List containing natural numbers

    Returns
    -------
    int
        The smallest natural number which is not in argument list.
    '''
    if 0 not in l:
        return 0
    else:
        num = 1

        while True:
            if (num not in l):
                return num
            else:
                num += 1

print("*** Program to Return Smallest Natural Number Not in List ***")

# Declare empty list
l = []
```

```
# Input
lLen = int(input("\nEnter length of list: "))

for i in range(lLen):
    l.append(int(input("Enter element: ")))

# Remove duplicates - only distinct numbers in list
l = list(set(l))

# Output
print("\nList:", l)
print("Note: Duplicates were removed.")

out = minout(l)    # Function call
print("\nOutput:", out)
```

RESULT:
The concepts related to list containers were studied.

INPUT / OUTPUT:

*** Program to Return Smallest Natural Number Not in List ***

Enter length of list: 5

Enter element: 1

Enter element: 3

Enter element: 2

Enter element: 4

Enter element: 17

List: [1, 2, 3, 4, 17]
Note: Duplicates were removed.

Output: 0

*** Program to Return Smallest Natural Number Not in List ***

Enter length of list: 5

Enter element: 1

Enter element: 3

Enter element: 0

Enter element: 2

Enter element: 4

List: [0, 1, 2, 3, 4]
Note: Duplicates were removed.

Output: 5

Ex. No: 8	USER-DEFINED FORMATTED OUTPUT
21 / 04 / 22	

AIM:

To learn about user-defined formatted output.

QUESTION:

Write a function to format the output of a float number which controls the number of digits and alignment. The function should be named as format_output() and takes two arguments, a float number x and a string which contains specifications for digits to print and alignment.

For example,
x = 1234.56789

format_output(x, '0.2f') should print '1234.56' (only 2 decimal places)

format_output(x, '>10.1f') should print ' 1234.5' (right justified, contains 4 blanks spaces followed by four digits and one decimal place)

format_output(x, '<10.1f') should print '1234.5 ' (contains four blank spaces at the end) Assume that the length of the formatted output never exceeds 10.

ALGORITHM:

Step 1: Read characters (total length of output string) in string until '.' (decimal point).
Read characters (number of decimal places) in string until 'f'.
Number of spaces = Total length of output string – Number of decimal places Number of digits in floating-point number to be formatted – 1.

Step 2: If first character of string is '<',
Print integral part of floating-point number 'x'.
Print '.' (decimal point) and floating-point digits.
Print spaces.
If first character of string is '>',
Print spaces.
Print integral part of floating-point number 'x'.
Print '.' (decimal point) and floating-point digits.
If first character of string is not '>' or '<',
Print integral part of floating-point number 'x'.
Print '.' (decimal point) and floating-point digits.

Program:

```
"""
Write a function to format the output of a float number which controls the
number of digits and alignment. The function should be named as
format_output()
and takes two arguments, a float number x and a string which contains
specifications for digits to print and alignment.


For example,
x = 1234.56789

format_output(x, ‘0.2f’) should print ‘1234.56’  (only 2 decimal places)

format_output(x, ‘>10.1f’) should print ‘    1234.5’ (right justified,
contains
4 blanks spaces followed by four digits and one decimal place)

format_output(x, ‘<10.1f’) should print ‘1234.5    ’ (contains four blank
spaces at the end)

Note: Assume that the length of the formatted output never exceeds 10.
"""

def format_output(x, string):
    '''
    Prints the formatted number 'x'.

    PARAMETERS
    ----------
    x : float
        Floating-point number to be formatted

    string : str
        Conditions to format output

    RETURNS
        None
    '''
    # Left or right justify
    if string[0] == '>' or string[0] == '<':
        totLen = []
        i = 1

        # Read total length of string
        while string[i] != '.':
            totLen.append(string[i])
            i += 1

        totLen = int("".join(totLen))
        i += 1
```

```
        # Read number of digits after '.' to be shown
        afterPointCount = []

        while string[i] != 'f':
            afterPointCount.append(string[i])
            i += 1

        afterPointCount = int("".join(afterPointCount))

        # Calculate spaces to insert before or after number (left / right
justify)
        spaces = totLen - afterPointCount - len(str(int(x))) - 1
        # Note: Subtract one space for decimal point

        # Left justify
        if (string[0] == '<'):
            # Print integral part of number
            print('\'', int(x), '.', sep="", end="")

            # Print floating-point part
            for i in range(2, afterPointCount + 2):
                print(str(x - int(x))[i], end="")

            # Print spaces after number
            for i in range(spaces):
                print(" ", end = "")

            print('\'')

        # Right justify
        else:
            print('\'', end="")

            # Print spaces before number
            for i in range(spaces):
                print(" ", end = "")

            # Print integral part of number
            print(int(x), '.', sep="", end="")

            # Print floating-point part
            for i in range(2, afterPointCount + 2):
                print(str(x - int(x))[i], end="")
            print('\'')

    else:
        # Read number of digits after '.' (decimal point) to be shown
        afterPointCount = []
        i = 2

        while string[i] != 'f':
```

```
            afterPointCount.append(string[i])
            i += 1

        afterPointCount = int("".join(afterPointCount))

        # Print integral part of number
        print('\'', int(x), '.', sep="", end="")

        # Print digits after decimal point (floating-point part)
        for i in range(2, afterPointCount + 2):
            print(str(x - int(x))[i], end="")

        print('\'')

x = 1234.56789

# Function call + Output
format_output(x, '0.2f')
format_output(x, '>10.1f')
format_output(x, '<10.1f')
```

RESULT:
The concepts related to user-defined formatted output were studied.

INPUT / OUTPUT:

'1234.56'
'   1234.5'
'1234.5   '

Ex. No: 9	IMPLEMENT STRING BUILT-IN METHOD
26 / 04 / 22	

AIM:

To learn to implement string build-in methods.

QUESTION 1:

Write a program to strip unwanted characters from strings based on specified conditions. The characters removed can be either from the beginning, end or both. These conditions are coded into individual functions as follows:

strip() method can be used to remove characters from the beginning and end of a string. If no arguments are provided, then the function removes whitespace, otherwise it removes the specified character as an argument.

lstrip() method removes whitespace or a specified character from the beginning of the string.

rstrip() method removes whitespace or a specified character from the end of the string.

Note:

1. If the character to be removed appears in between other characters, then nothing should be removed.

2. All the methods take the string as its first argument. Examples:

s1 = ‘ hello world ’
s2 = ‘---==hello world===---’

strip(s1) should output ‘hello world’ (the whitespace between hello and world is not removed because it appears in between)

strip(s2, ‘-’) should output ‘==hello world===’

lstrip(s1) should output ‘hello world ’
lstrip(s2, ‘-’) should output ‘==hello world===---’
lstrip(s2, ‘=’) should output ‘---==hello world===---’ (because s2 is not beginning with ‘=’)
rstrip(s1) should output ‘ hello world’
rstrip(s2, ‘-’) should output ‘---==hello world===’
rstrip(s2, ‘=’) should output ‘---==hello world===---’ (because s2 is not ending with ‘=’)

ALGORITHM:

lstrip( )

Step 1: i = 0
Step 2: From first character in string (or) i = 0 (starting from beginning),
    If character ‘string[i]’ in ‘chars’ (‘chars’ - characters to be stripped),
        Move to next character in string (or) increment ‘i’ by ‘1’.
    Else,
        Return string from ‘i^{th}’ index till end.

rstrip( )

Step 1: i = length of string - 1
Step 2: From last character in (starting from end),
    If character ‘string[i]’ in ‘chars’ (‘chars’ - characters to be stripped),
        Move to next character (towards left) in string (or) decrement ‘i’ by ‘1’.
    Else,
        Return string from start till ‘i^{th}’ index (inclusive).

strip( )

Step 1: Call ‘lstrip()’.
Step 2: Call ‘rstrip()’.

Program:

```
"""
Write a program to strip unwanted characters from strings based on
specified conditions.
The characters removed can be either from the beginning, end, or both.
These conditions are
coded into individual functions as follows:

- strip() method can be used to remove characters from the beginning and
end of a string.
If no arguments are provided, then the function removes whitespace,
otherwise it removes
the specified character as an argument.

- lstrip() method removes whitespace or a specified character from the
beginning of the string.

- rstrip() method removes whitespace or a specified character from the end
of the string.

Note:
If the character to be removed appears in between other characters, then
nothing should be removed.
All the methods take the string as its first argument.

Examples:
s1 = ‘    hello world    ’
s2 = ‘---==hello world===---’

strip(s1) should output ‘hello world’ (the whitespace between hello and
world is not removed because it appears in between)
strip(s2, ‘-’) should output ‘==hello world===’

lstrip(s1) should output ‘hello world    ’
lstrip(s2, ‘-’) should output ‘==hello world===---’
lstrip(s2, ‘=’) should output ‘---==hello world===---’ (because s2 is not
beginning with ‘=’)

rstrip(s1) should output ‘    hello world’
rstrip(s2, ‘-’) should output ‘---==hello world===’
rstrip(s2, ‘=’) should output ‘---==hello world===---’ (because s2 is not
ending with ‘=’)
"""

def lstrip(string, chars = ' '):
    '''
    Returns string with chars stripped on the left.
    Default chars is whitespace.

    Parameters
    ----------
```

```
    string : str
        Input string.
    chars : str, optional
        Character(s) to strip. The default is ' '.

    Returns
    -------
    str
        String with chars stripped on the left.
    '''
    i = 0

    while string[i] in chars:
        i += 1

    return string[i:]

def rstrip(string, chars = ' '):
    '''
    Returns string with chars stripped on the right.
    Default chars is whitespace.

    Parameters
    ----------
    string : str
        Input string.
    chars : str, optional
        Character(s) to strip. The default is ' '.

    Returns
    -------
    str
        String with chars stripped on the right.
    '''
    i = len(string) - 1

    while string[i] in chars:
        i -= 1

    return string[:i+1]

def strip(string, chars = ' '):
    '''
    Returns string with chars stripped on both sides.
    Default chars is whitespace.

    Parameters
    ----------
    string : str
        Input string.
    chars : str, optional
        Character(s) to strip. The default is ' '.
```

```
    Returns
    -------
    str
        String with chars stripped on both sides.
    '''
    string = lstrip(string, chars)
    string = rstrip(string, chars)

    return string

# Strings
s1 = '    hello world    '
s2 = '---==hello world===---'

# strip() - Output
print(strip(s1))
print(strip(s2, '-'))
print()

# lstrip() - Output
print(lstrip(s1))
print(lstrip(s2, '-'))
print(lstrip(s2, '='))
print()

# rstrip() - Output
print(rstrip(s1))
print(rstrip(s2, '-'))
print(rstrip(s2, '='))
```

RESULT:
The concepts related to string built-in methods were studied.

INPUT / OUTPUT:

hello world
==hello world===

hello world
==hello world===---
---==hello world===---

    hello world
---==hello world===---

Ex. No: 10	DICTIONARY CONTAINER
05 / 05 / 22	

AIM:

To learn about dictionary containers.

QUESTION 1:

Create a dictionary from two lists as shown below:

key = [‘a’,’b’,’c’,’d’,’e’,’f’]

value = [1,2,3,4,5,6]

D = {‘a’:1,’b’:2,’c’:3,’d’:4,’e’:5,’f’:6]

Check if the length of key and value are the same before performing the operation. If not, print a message mentioning the dictionary cannot be created due to length mismatch.

ALGORITHM:

Step 1: Declare and initialise key list.
Step 2: Declare and initialise value list.

Step 3: If lengths of both lists are not equal,
Print error message.
Else,
Create an empty dictionary (output dictionary).
For every key in key list,
Append key to dictionary with value from value dictionary (according to index).
Step 4: Print output dictionary.

QUESTION 2:

Create a word count dictionary using a string as shown below: i/p S = ‘the quick brown fox jumps on the lazy dog’

o/p: {'t': 2, 'h': 2, 'e': 2, ' ': 8, 'q': 1, 'u': 2, 'i': 1, 'c': 1, 'k': 1, 'b': 1, 'r': 1, 'o': 4, 'w': 1, 'n': 2, 'f': 1, 'x': 1, 'j': 1, 'm': 1, 'p': 1, 's': 1, 'l': 1, 'a': 1, 'z': 1, 'y': 1, 'd': 1, 'g': 1}

ALGORITHM:

Step 1: Create a character count dictionary.
Step 2: For every character in list,
    If character is in dictionary,
        Increment count (value) by 1.
    Else,
        Add character as a key in dictionary and set value to '0',
Step 3: Print the character count dictionary as output.

Program 1:

```
"""
Create a dictionary from two lists as shown below:

key = [‘a’, ’b’, ’c’, ’d’ ,’e’, ’f’]
value = [1, 2 , 3 , 4 , 5 , 6]

D = {‘a’ : 1, ’b’ : 2, ’c’ : 3, ’d’ : 4, ’e’ : 5, ’f’ : 6]

Check if the length of key and value are the same before performing
the operation. If not, print a message mentioning the dictionary
cannot be created due to length mismatch.
"""

key = ['a', 'b', 'c', 'd', 'e', 'f']
value = [1, 2 , 3 , 4 , 5 , 6]

if len(key) != len(value):
    print("Dictionary cannot be created due to length mismatch.")
else:
    # Declare an empty dictionary
    D = {}

    # Create 'key : value' pairs
    for i in range(len(key)):
        D[key[i]] = value[i]

    # Output
    print(D)
```

INPUT / OUTPUT:

{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6}

Program 2:

```
"""
Create a character count dictionary using a string as shown below:

i/p: S = ‘the quick brown fox jumps on the lazy dog’
o/p: {'t': 2, 'h': 2, 'e': 2, ' ': 8, 'q': 1, 'u': 2, 'i': 1,
      'c': 1, 'k': 1, 'b': 1, 'r': 1, 'o': 4, 'w': 1, 'n': 2,
      'f': 1, 'x': 1, 'j': 1, 'm': 1, 'p': 1, 's': 1, 'l': 1,
      'a': 1, 'z': 1, 'y': 1, 'd': 1, 'g': 1}
"""

# String
S = 'the quick brown fox jumps on the lazy dog'

# Character count dictionary
charCount = {}

# Calculate character count
for char in S:
    if char in charCount:
        charCount[char] += 1
    else:
        charCount[char] = 1

# Output
print(charCount)
```

RESULT:

The concepts related to dictionary containers were studied.

INPUT / OUTPUT:

{'t': 2, 'h': 2, 'e': 2, ' ': 8, 'q': 1, 'u': 2, 'i': 1, 'c': 1, 'k': 1, 'b': 1, 'r': 1, 'o': 4, 'w': 1, 'n': 2, 'f': 1, 'x': 1, 'j': 1, 'm': 1, 'p': 1, 'j': 1, 'm': 1, 'p': 1, 's': 1, 'l': 1, 'a': 1, 'z': 1, 'y': 1, 'd': 1, 'g': 1}