**Experiment No 1:**

**Aim:** To implement the basic data types and control structures in python.

**Theory:**

Python has the following data types built-in by default, in these categories

Text Type: Str

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Mapping Type: Dict

Set Types: set, frozenset

Boolean Type: Bool

Binary Types: bytes, bytearray, memoryview

**Getting the Data Type**

You can get the data type of any object by using the type() function:

Print the data type of the variable x:

```
x = 5
print(type(x))
```

**Casting**

There can be two types of Type Casting in Python –

- Implicit Type Casting
- Explicit Type Casting

**Implicit Type Conversion**

In this, methods, Python converts data type into another data type automatically. In this process, users don't have to involve in this process.

```
# Python program to demonstrate
# implicit type Casting
 # Python automatically converts
# a to int
```

```
a = 7
print(type(a))

# Python automatically converts
# b to float
b = 3.0
print(type(b))

# Python automatically converts
# c to float as it is a float addition
c = 0.5 + 0.5
print(c)
print(type(c))

# Python automatically converts
# d to float as it is a float multiplication
d = 0.5 * 0.5
print(d)
print(type(d))
```

Output:


**Explicit Type Casting**

In this method, Python need user involvement to convert the variable data type into certain data type in order to the operation required.

Mainly in type casting can be done with these data type function:

- **Int() :** Int() function take float or string as an argument and return int type object.
- **float() :** float() function take int or string as an argument and return float type object.
- **str() :** str() function take float or int as an argument and return string type object.

**Let's see some example of type casting:**

**Type Casting int to float:**

Here, we are casting integer object to float object with **float()** function.

```
# Python program to demonstrate
# type Casting

# int variable
a = 5
```

```
 # typecast to float
n = float(a)
print(n)
print(type(n))
```

**Output:**

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\explicit.py"
5.0
<class 'float'>
PS C:\Users\Lenovo\Downloads\Python Prgs>
```



**Sequence data types**

Python has 4 built in in data types used to store collections of data, the List,Tuple, Set, and Dictionary, all with different qualities and usage.

**1.List:**Lists are used to store multiple items in a single variable.

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

**Output:**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR                    Code + ~

PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\list.py"
['apple', 'banana', 'cherry']
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**2.Tuple:**A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

**Output:**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR                    Code + ~

PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\tuples.py"
('apple', 'banana', 'cherry')
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**3.Set:**A set is a collection which is *unordered*, *unchangeable**, and *unindexed*. ***
**Note:** Set *items* are unchangeable, but you can remove items and add new items.
Sets are written with curly brackets.

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

**Output:**

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\set.py"
{'apple', 'banana', 'cherry'}
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**4.Dictionary:**A dictionary is a collection which is ordered*, changeable and do not allow duplicates.Dictionaries are written with curly brackets, and have keys and values:

```
thisdict = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}
print(thisdict)
```

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\dict.py"
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

```
# Python3 program for explaining
# use of list, tuple, set and
# dictionary

# Lists
l = []
 # Adding Element into list
l.append(5)
l.append(10)
print("Adding 5 and 10 in list", l)
# Popping Elements from list
l.pop()
print("Popped one element from list", l)
print()


# Set
s = set()
 # Adding element into set
s.add(5)
s.add(10)
```

```
print("Adding 5 and 10 in set", s)
# Removing element from set
s.remove(5)
print("Removing 5 from set", s)
print()



 # Tuple
t = tuple(l)
 # Tuples are immutable
print("Tuple", t)
print()


 # Dictionary
d = {}
 # Adding the key value pair
d[5] = "Five"
d[10] = "Ten"
print("Dictionary", d)



 # Removing key-value pair
del d[10]
print("Dictionary", d)
```

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\all.py"
Adding 5 and 10 in list [10]
Popped one element from list [10]

Adding 5 and 10 in set {10}
Removing 5 from set {10}

Tuple (10,)

Dictionary {10: 'Ten'}
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**Control Structures in Python**

Python programming language provides following types of loops to handle looping requirements.

**1.While Loop**

Syntax :
while expression:

statement(s)

i = 0

while i < 10:

    print(i)

    i += 1

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\while.py"
0
1
2
3
4
5
6
7
8
9
PS C:\Users\Lenovo\Downloads\Python Prgs>
```



**2.For in Loop**

Syntax:
for iterator_var in sequence:

statements(s)

for i in range(10):

    print(i)

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\for.py"
0
1
2
3
4
5
6
7
8
9
PS C:\Users\Lenovo\Downloads\Python Prgs>
```



**3.Nested Loops**

Syntax:
for iterator_var in sequence:

for iterator_var in sequence:

statements(s)

statements(s)

The syntax for a nested while loop statement in Python programming language is as follows:

```
while expression:
 while expression:
 statement(s)
 statement(s)
```

```
for i in
range(3):
    for j in
range(3):
        print(i,
j)

i = 0
j = 0

while i < 3:
    while j <
3:
        print(i,
j)
        j += 1
    i += 1
    j = 0
```

Output:

```
2 0
2 1
2 2
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**Control Statements**

**1.Continue Statement**

It returns the control to the beginning of the loop.

```
for i in range(0, 10):
    if i == 5:
        continue
    print(i)
    if i == 8:
        break
```

Ouptut:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\continue.py"
0
1
2
3
4
6
7
8
PS C:\Users\Lenovo\Downloads\Python Prgs>
```



**2. Break Statement**

It brings control out of the loop

```
for i in range(0,10):
 if (i==5):
 break
 print (i)
```

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\break.py"
5
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**2.Pass Statement**

We use pass statement to write empty loops. Pass is also used for empty control statement, function and classes.

```
for i in range(0, 10):
    if i == 5:
        pass
    print(i)
```

Output:

```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\pass.py"
0
1
2
3
4
5
6
7
8
9
PS C:\Users\Lenovo\Downloads\Python Prgs>
```

**PROGRAM:**

```
print("-----Program for Student Information-----")

D = dict()

n = int(input('How many student records do you want to store? '))

for i in range(n):
    x, y = input("Enter the complete name (First and last name) of the student:
").split()
    z = input("Enter contact number: ")
    m = input('Enter Marks: ')
    D[x, y] = (z, m)


# Define a function for sorting names based on first name
# Define the dictionary
D = {
    "John Doe": ["1234567890", 85],
    "Alice Smith": ["9876543210", 90],
    "Bob Johnson": ["5555555555", 80]
}

# Define the sort function
def sort(D):
    ls = []
    for sname, _ in D.items():
        # Splitting the name into first and last name
        first_name, last_name = sname.split()
        tup = (first_name, last_name)
        ls.append(tup)
    ls = sorted(ls)
    for i in ls:
        print(i[0], i[1])

# Define the minmarks function
```

```
def minmarks(D):
    marks_list = [details[1] for _, details in D.items()]
    print("Minimum marks:", min(marks_list))

# Define the searchdetail function
def searchdetail(D, fname):
    for sname, details in D.items():
        if sname.split()[0] == fname:
            print("Contact number:", details[0])
            return

# Define the option function
def option():
    choice = int(input('Enter the operation detail:\n1: Sorting using first name\n2:
Finding Minimum marks\n3: Search contact number using first name\n4: Exit\nOption: '))
    if choice == 1:
        sort(D)
    elif choice == 2:
        minmarks(D)
    elif choice == 3:
        first = input('Enter first name of student: ')
        searchdetail(D, first)
    elif choice == 4:
        print('Thanks for executing me!!!!')
        exit()
    else:
        print('Invalid option!')
        option()

# Main loop
while True:
    option()
    inp = input('Want to perform some other operation? (Y/N): ')
    if inp.upper() != 'Y':
        break
```

**Output :**

```
How many student records do you want to store? 2
Enter the complete name (First and last name) of the student: Ankit Bari
Enter contact number: 9239328923
Enter Marks: 100
Enter the complete name (First and last name) of the student: Yash Kerkar
Enter contact number: 8888748225
Enter Marks: 80
Enter the operation detail:
1: Sorting using first name
2: Finding Minimum marks
3: Search contact number using first name
4: Exit
Option: 2
Minimum marks: 80
Want to perform some other operation? (Y/N): Y
Enter the operation detail:
1: Sorting using first name
2: Finding Minimum marks
3: Search contact number using first name
4: Exit
Option: 3
Enter first name of student: Yash
Want to perform some other operation? (Y/N): Y
Enter the operation detail:
1: Sorting using first name
2: Finding Minimum marks
3: Search contact number using first name
4: Exit
Option: 4
Thanks for executing me!!!!
```

**Conclusion:** the experiment effectively showcased the integration of essential data types and control structures within Python. By engaging in practical exercises, participants acquired proficiency in manipulating variables, employing loops, leveraging conditionals, and utilizing fundamental data structures