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

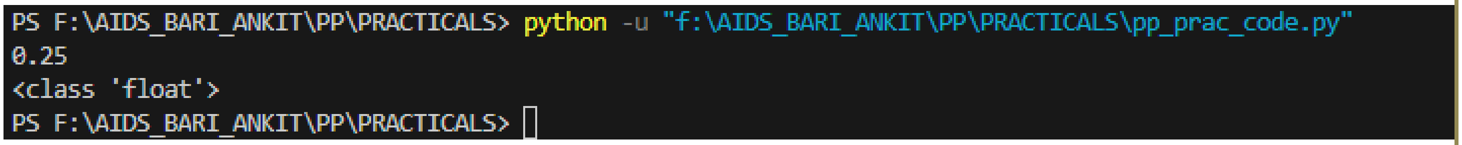
Implicit Type Conversion

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

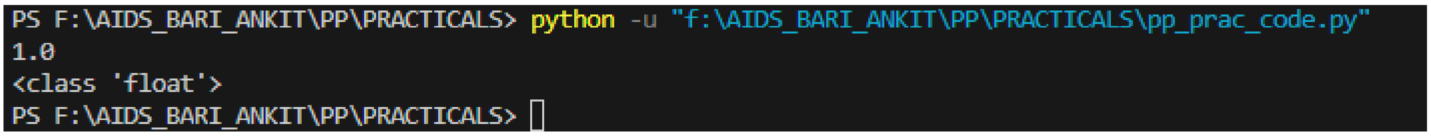
```
# Python program to demonstrate
# implicit type Casting
 # Python automatically converts
# a to int
```

```
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))
```

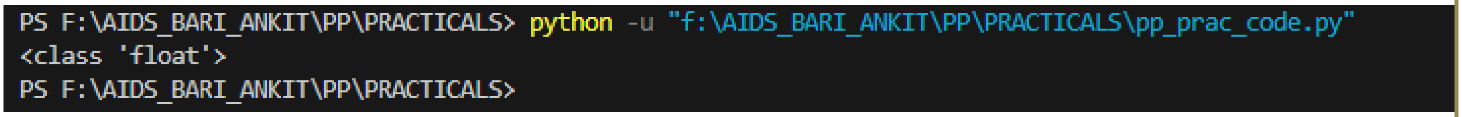
Output:



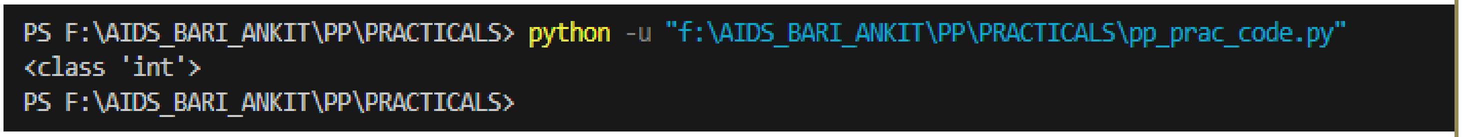
```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0.25
<class 'float'>
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
1.0
<class 'float'>
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
<class 'float'>
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
<class 'int'>
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

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

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

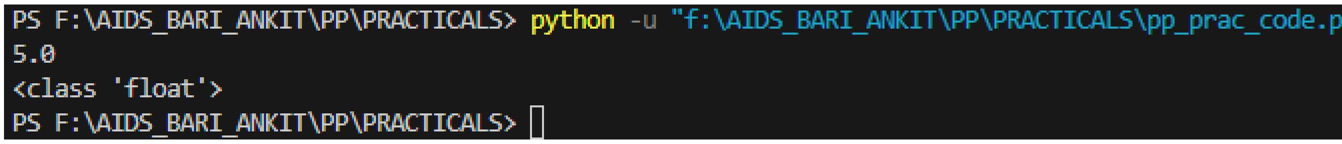
- **Int() :** Int() function take float or string as an argument and return int type object.
- **float() :** float() function take int or string as an argument and return float type object.
- **str() :** str() function take float or int as an argument and return string type object.

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

#
i
n
t
v
a
r
i
a
b
l
e
a
=
5
 #
ty
pe
ca
st
to
fl
oa
tn
=
fl
oa
t(
a)
pr
in
t(
n)
pr
in
t(t
y
pe
(n
))
```

Output:



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.p
5.0
<class 'float'>
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

5.0

<class 'float'>

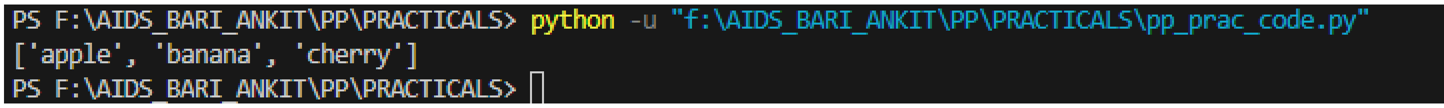
Sequence data types

 Python has 4 built in in data types used to store collections of data, the List,Tuple, Set, andDictionary, 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)



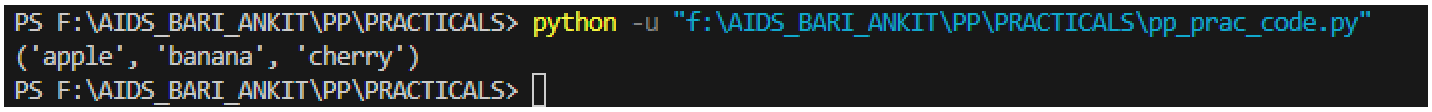
```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
['apple', 'banana', 'cherry']
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

2.Tuple:A tuple is a collection which is ordered

and **unchangeable**.Tuples are written with round

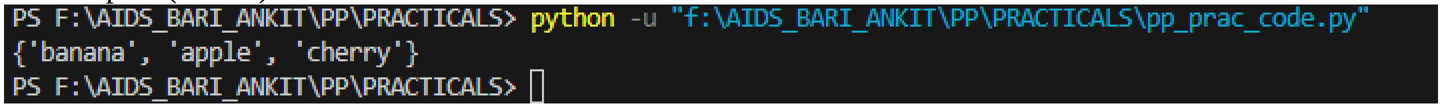
brackets.

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



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
('apple', 'banana', 'cherry')
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

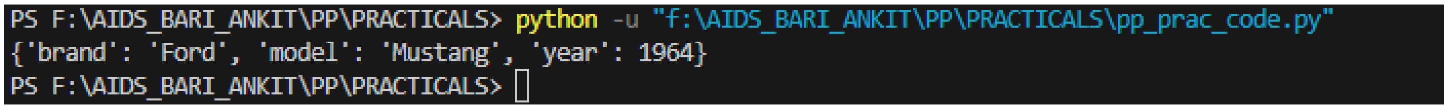
3.Set:A set is a collection which is *unordered, unchangeable**, and *unindexed*. ***Note:** Set *items* are unchangeable, but you can remove items and add new items. Sets are written with curly brackets.
thisset = {"apple",
"banana", "cherry"}
print(thisset)



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
{'banana', 'apple', 'cherry'}
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

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

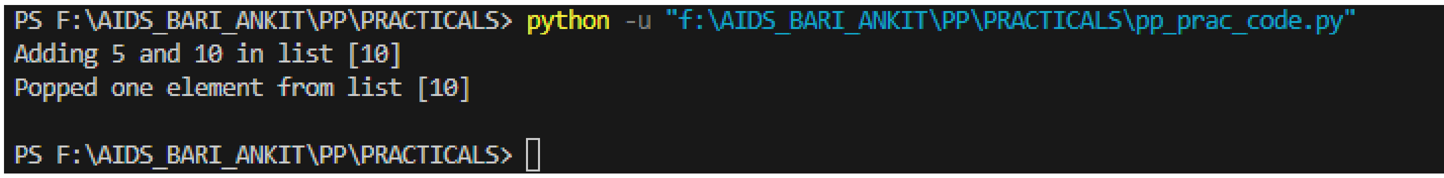
```
 t
h
i
s
d
i
c
t
=
{
"
b
r
a
n
d
"
:
"
F
o
r
d
"

,
"mo
del":
"Mu
stan
g",
"yea
r":
1964
}
print(thisdict)
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

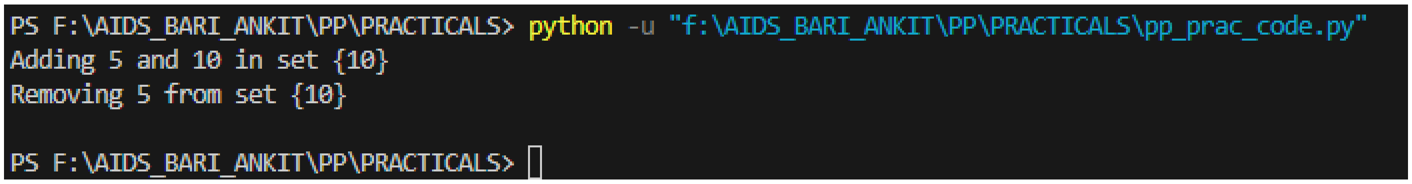
```
# Python3
program for
explaining# use
of list, tuple, set
and
# dictionary

#
L
i
s
t
s
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)
p
 r
 i
```

n
t
(
)



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Adding 5 and 10 in list [10]
Popped one element from list [10]

PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

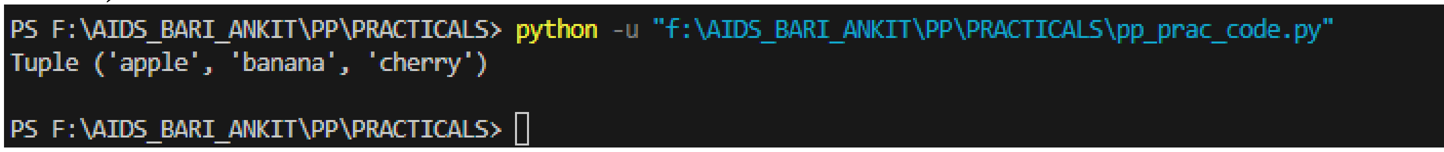
#
S
e
t
s = set()
# Adding
element
into set
s.add(5)
s.add(10)
print("Adding 5
and 10 in set", s)#
Removing
element from set
s.remove(5)
print("Removin
g 5 from set", s)
print()



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Adding 5 and 10 in set {10}
Removing 5 from set {10}

PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

# Tuple
t = tuple(l)
# Tuples are immutable
p
r
i
n
t

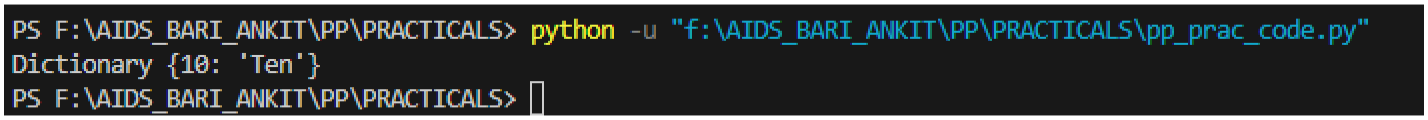
(
"
T
u
p
l
e
"
,
t
)
p
r
i
n
t
(
)



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Tuple ('apple', 'banana', 'cherry')

PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

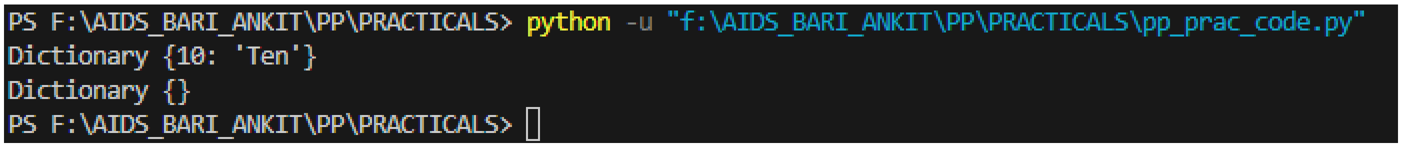
#
D
i
c
t
i
o
n
a
r
y

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



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Dictionary {10: 'Ten'}
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

 #
Removing
key-value
pair**del**
d[10]
print("Dicti
onary", d)



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Dictionary {10: 'Ten'}
Dictionary {}
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

Control Structures in Python

Python programming language provides following types of loops
to handle loopingrequirements.

1.While Loop

Syntax :
while expression:

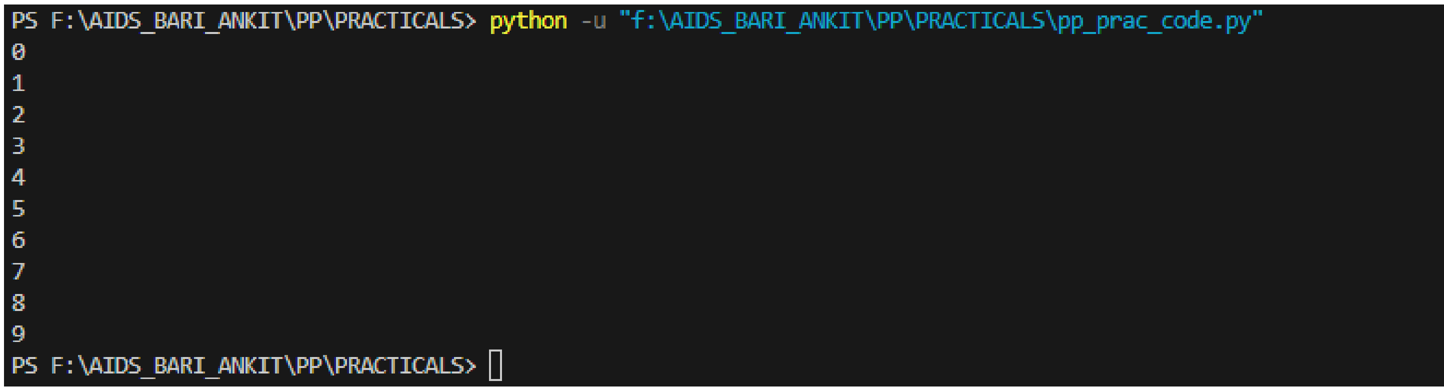
 statement(s)

 i = 0

 while i < 10:

    print(i)

```
    i += 1
```

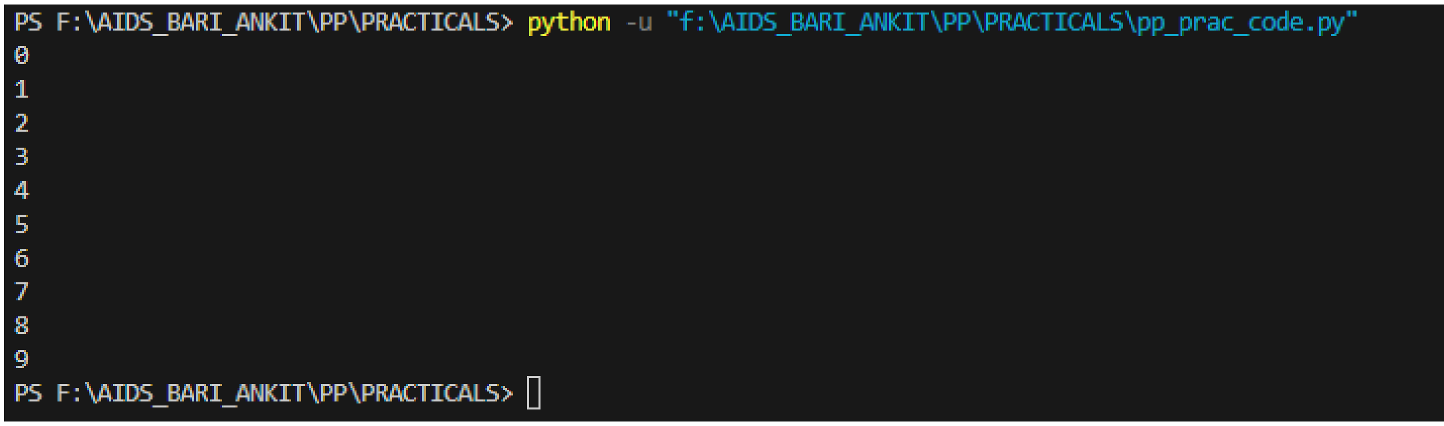


```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0
1
2
3
4
5
6
7
8
9
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

2.For in Loop

Syntax:

```
for iterator_var in sequence:
 statements(s)
 for i in range(10):
    print(i)
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0
1
2
3
4
5
6
7
8
9
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

3.Nested Loops

Syntax:

```
for iterator_var in sequence:
 for iterator_var in sequence:
 s
```

t

a

t

e

m

e

n

t

s

(

s

)

s

t

a

t

e

m

e

n

t

s

(

s

)

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

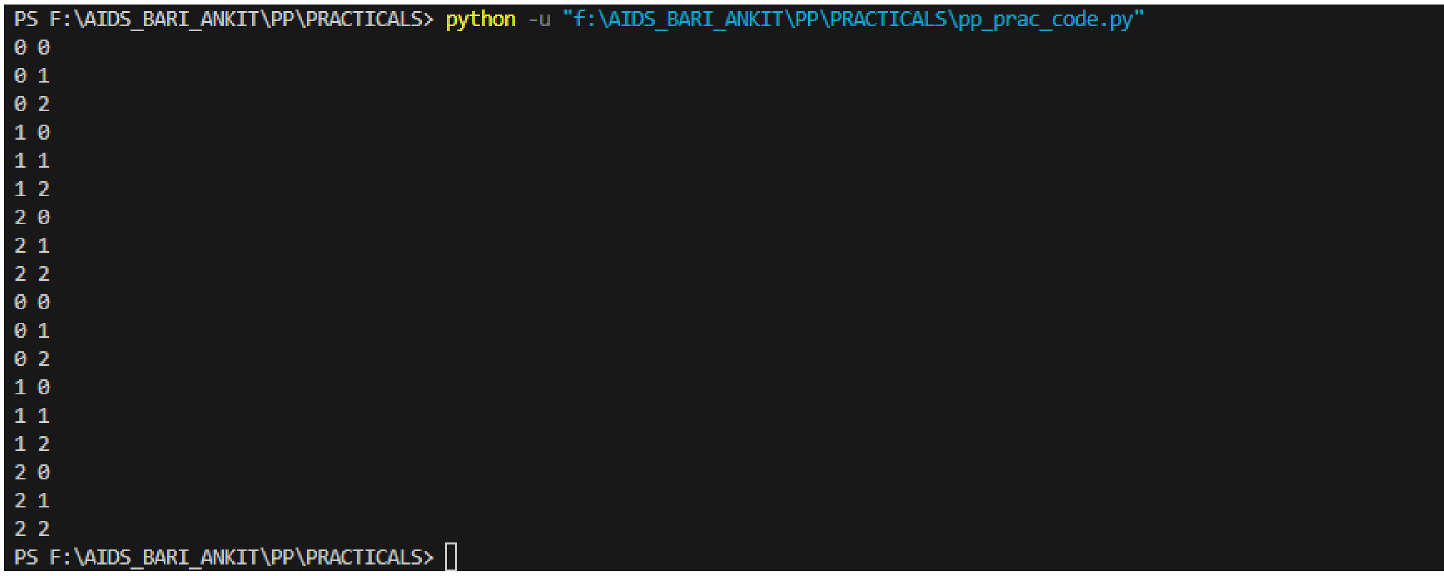
while expression:
while expression:
s
t
a
t

e
m
e
n
t
(
s
)
s
t
a
t
e
m
e
n
t
(
s
)
f
o
r
i
i
n
r
a
n
g
e
(
3
)
:

f
o
r
j
i
n
r
a
n
g

```
e
(
3
)
:


p
r
i
n
t
(
i
,
j
)

i
=
0
j
=
0

w
h
i
l
e
i
<
3
:

w
h
i
l
e
j
<
3
:
```

```
p
r
i
n
t
(
i
,
j
)


j
+
=
1

i
+
=
1

j
=
0
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0 0
0 1
0 2
1 0
1 1
1 2
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 F:\AIDS_BARI_ANKIT\PP\PRACTICALS> 
```

Contr

ol

State

ments

1.Cont

inue

State

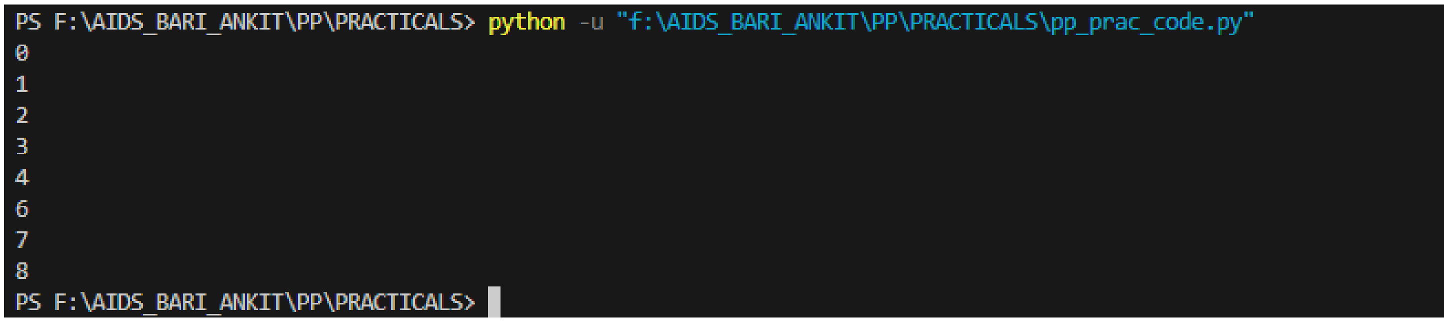
ment

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
```



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0
1
2
3
4
6
7
8
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

2. Break Statement

It brings control

out of the loop

for i in

range(0,10):

if (i==5):

b

r

e

a

k

p

r

i

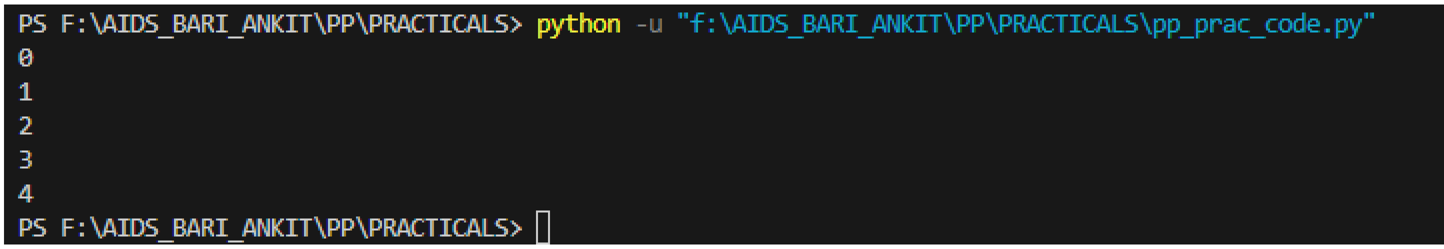
n

t

(

i

)

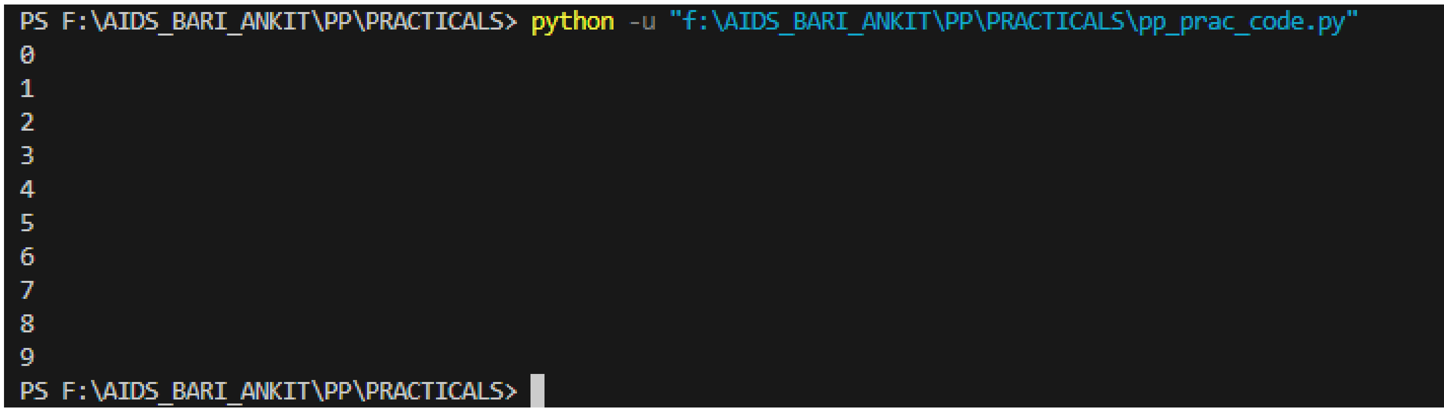


```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0
1
2
3
4
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

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)
```



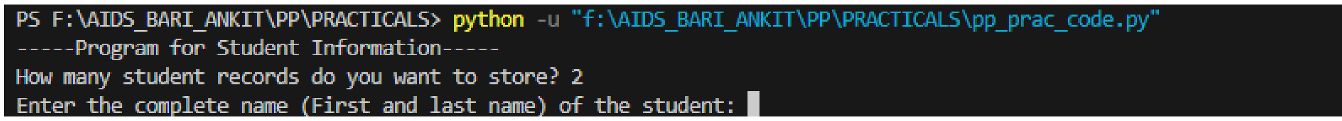
```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
0
1
2
3
4
5
6
7
8
9
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS>
```

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
def sort():
    ls = list()
    # Fetch key and value using items() method
    for sname, details in D.items():
        # Store key parts as a tuple
        tup = (sname[0], sname[1])
        # Add tuple to the list
        ls.append(tup)
    # Sort the final list of tuples
    ls = sorted(ls)
    for i in ls:
        # Print first name and second name
        print(i[0], i[1])

# Define a function for finding the minimum marks in stored data
def minmarks():
    ls = list()
    # Fetch key and value using items() method
    for sname, details in D.items():
        # Add details second element (marks) to the list
        ls.append(details[1])
    # Sort the list elements
    ls = sorted(ls)
    print("Minimum marks:", min(ls))

# Define a function for searching student contact number
def searchdetail(fname):
    for sname, details in D.items():
        if sname[0] == fname:
            print(details[0])
            return



# Define a function for asking the options
def option():
    choice = int(input('Enter the operation detail: \n 1: Sorting using first name \n 2:
Finding Minimum marks \n 3: Search contact number using first name \n 4: Exit\n
Option: '))
    if choice == 1:
        # Function call sort()
        sort()
    elif choice == 2:
        minmarks()
    elif choice == 3:
        first = input('Enter first name of student: ')
        searchdetail(first)
    elif choice == 4:
        print('Thanks for executing me!!!!')
        exit()
    else:
        print('Invalid option!')
        option()


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

Output :



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
-----Program for Student Information-----
How many student records do you want to store? 2
Enter the complete name (First and last name) of the student:
```

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 utilizingfundamental data structures.