Lab Exercises to Practice

```
# 1
var_x= input("Enter a signed value ")
print(var_x)
```

```
Enter a signed value -10
-10
```

```
# 2

var_octal=0o435247
print(var_octal)

var_hex=0x79826AB
print(var_hex)
```

```
146087
127411883
```

+ Code + Text

```
# 3 float
height=5.38
type(height)
```

```
float
```

```
# 4 complex variable
a= 10
b= 4
c= complex(a,b)
print(c)
type(c)
```

```
(10+4j)
complex
```

```
# 5
i, var_long, f, str1= 10, 299749, 34.28, "Sneha"
print(i)
print(var_long)
print(f)
print(str1)
```

```
10
299749
34.28
Sneha
```

```
# 6
var_float= 36.95
print(var_float)
del var_float
```

```
36.95
```

```
# 7
# #! symbol indicates the interpreter, or which version of an interpreter, to use when executing a script.
```

```
# 8
var_str1= "Sneha Bhadauria"
print(var_str1)
print(var_str1[0])
print(var_str1[2:5])
print(var_str1[3:])
print(var_str1 * 2)
str_2= "Welcome!"
print(var_str1+" " +str_2)
```

```
Sneha Bhadauria
S
```

```
eha
ha Bhadauria
Sneha BhadauriaSneha Bhadauria
Sneha Bhadauria Welcome!
```

```
# 9 Lists
var_list= ["Apple","Banana","Kiwi","Guava"]
print(var_list)
print(var_list[0])
print(var_list[1:3])
print(var_list[2:])
print(var_list * 2)
var_list2=["Mango","Litchi"]
print(var_list + ["Plum"])
print(var_list + var_list2)
```

```
['Apple', 'Banana', 'Kiwi', 'Guava']
Apple
['Banana', 'Kiwi']
['Kiwi', 'Guava']
['Apple', 'Banana', 'Kiwi', 'Guava', 'Apple', 'Banana', 'Kiwi', 'Guava']
['Apple', 'Banana', 'Kiwi', 'Guava', 'Plum']
['Apple', 'Banana', 'Kiwi', 'Guava', 'Mango', 'Litchi']
```

```
# 10 Tuple

var_tuple= ("Apple","Banana","Cherry")
print(var_tuple)
print(var_tuple[0])
print(var_tuple[1:3])
print(var_tuple[2:])
print(var_tuple[1:3])
print(var_tuple * 2)
print(var_tuple+("Dragonfruit","Orange"))
var_tuple2=("Mango","Berry")
print(var_tuple+var_tuple2)
```

```
('Apple', 'Banana', 'Cherry')
Apple
('Banana', 'Cherry')
('Cherry',)
('Banana', 'Cherry')
('Apple', 'Banana', 'Cherry', 'Apple', 'Banana', 'Cherry')
('Apple', 'Banana', 'Cherry', 'Dragonfruit', 'Orange')
('Apple', 'Banana', 'Cherry', 'Mango', 'Berry')
```

```
# 11 Dictionary
var_dict={'One':'Sneha',
          'Two':'Karan',
          'Three':'Abhijith'}
var_dict[2]="This is two"
print(var_dict['One'])
print(var_dict[2])
print(var_dict)
print(var_dict.keys())
print(var_dict.values())
```

```
Sneha
This is two
{'One': 'Sneha', 'Two': 'Karan', 'Three': 'Abhijith', 2: 'This is two'}
dict_keys(['One', 'Two', 'Three', 2])
dict_values(['Sneha', 'Karan', 'Abhijith', 'This is two'])
```

```
# 12 Set

set1=set({})

print(type(set1))
```

```
<class 'set'>
```

```
# 13 List of Sets

list_of_set=[{1,2,3},{"Sneha","Karan"}]
print(list_of_set)
print(type(list_of_set))
```

```
[{1, 2, 3}, {'Sneha', 'Karan'}]
<class 'list'>
```

```
# 14 List of Tuples

list_of_tuple=[(1,2,3),("Sneha","Ashish","Karan")]
print(list_of_tuple)
print(type(list_of_tuple))
```

```
[(1, 2, 3), ('Sneha', 'Ashish', 'Karan')]
<class 'list'>
```

```
# 15 List Of Dictionatries

list_of_dict=[{'One':'Sneha','Two':'Karan'},{1:"This is One",2:"This is two"}]
print(list_of_dict)
```

```
[{'One': 'Sneha', 'Two': 'Karan'}, {1: 'This is One', 2: 'This is two'}]
```

```
# 16 List of Sets and Tuple

list_of_set_tuple =[{1,2,3},("Sneha","Karan","Abhijith")]
print(list_of_set_tuple)
```

```
[{1, 2, 3}, ('Sneha', 'Karan', 'Abhijith')]
```

```
# 17 List of dict,Sets and Tuple

list_of_dict_set_tuple =[{1:"This is One",2:"This is two"},{1,2,3},("Sneha","Karan","Abhijith")]
print(list_of_dict_set_tuple)
```

```
[{1: 'This is One', 2: 'This is two'}, {1, 2, 3}, ('Sneha', 'Karan', 'Abhijith')]
```

18 Diff btw list, Set, Tuple and Dictionary

A list is a non-homogeneous data structure that stores the elements in columns of a single row or multiple rows.

A Tuple is also a non-homogeneous data structure that stores elements in columns of a single row or multiple rows.

The set data structure is also a non-homogeneous data structure but stores the elements in a single row.

A dictionary is also a non-homogeneous data structure that stores key-value pairs. /----------/

The list can be represented by [ ]

Tuple can be represented by ( )

The set can be represented by { }

The dictionary can be represented by { }

/----------/

The list allows duplicate elements

Tuple allows duplicate elements

The Set will not allow duplicate elements

The dictionary doesn’t allow duplicate keys.

/----------/

The list can use nested among all

Tuple can use nested among all

The set can use nested among all

The dictionary can use nested among all

/----------/ Example: [1, 2, 3, 4, 5]

Example: (1, 2, 3, 4, 5)

Example: {1, 2, 3, 4, 5}

Example: {1: “a”, 2: “b”, 3: “c”, 4: “d”, 5: “e”}

/----------/ A list can be created using the list() function

Tuple can be created using the tuple() function.

A setA dictionary can be created using the set() function

A dictionary can be created using the dict() function.

/----------/

A list is mutable i.e we can make any changes in the list.

A tuple is immutable i.e we can not make any changes in the tuple.

A set is mutable i.e we can make any changes in the set, ut elements are not duplicated.

A dictionary is mutable, ut Keys are not duplicated.

/----------/

List is ordered

Tuple is ordered

Set is unordered

Dictionary is ordered (Python 3.7 and above)

/----------/

Creating an empty list

l=[]

Creating an empty Tuple

t=()

Creating a set

a=set() b=set(a)

Creating an empty dictionary

d={}

/----------/

19

Lists and tuples are ordered and allow duplicate elements, but lists are mutable, and tuples are immutable. Sets are unordered and do not allow duplicate elements. Dictionaries are collections of key-value pairs, and keys must be unique.

```
# 20 Type Conversion
a =83
b=[1,2,3]
var_int=int(a)
var_float=float(a)
var_tuple=tuple(b)
var_list=list(b)
var_dict=dict({a:b})
print(type(var_int))
print(type(var_float))
print(type(var_tuple))
print(type(var_list))
print(type(var_dict))
var_char=chr(a)
print(type(var_char))

var_string=str(345)
print(type(var_string))
```

```
<class 'int'>
<class 'float'>
<class 'tuple'>
<class 'list'>
<class 'dict'>
<class 'str'>
<class 'str'>
```

```
# 21
sales_quantity= 100
unit_price= 50
sales_amount=sales_quantity*unit_price
if sales_amount > 1000:
  print('Good')
if sales_amount >500 and sales_amount < 1000:
  print("Average")
if sales_amount < 500:
  print("Poor")
```

```
Good
```

```
# 22
DOB=int(input("Enter Year of Birth: "))
current_year=2023
age=current_year-DOB
if age>100:
  print("Above 100")
elif age>75 and age<=100:
  print("76-100")
elif age>50 and age<=75:
  print("50-75")
elif age>25 and age<=50:
  print("25-50")
else:
  print("Under 25")
```

```
Enter Year of Birth: 2000
Under 25
```

23

For loop is used to iterate over a sequence of items. While loop is used to repeatedly execute a block of statements while a condition is true.

For loops are designed for iterating over a sequence of items. Eg. list, tuple, etc.

While loop is used when the number of iterations is not known in advance or when we want to repeat a block of code until a certain condition is met. For loop require a sequence to iterate over.

While the loop requires an initial condition that is tested at the beginning of the loop.

For loop is typically used for iterating over a fixed sequence of items While loop is used for more complex control flow situations.

For loop is more efficient than a while loop when iterating over sequences, since the number of iterations is predetermined and the loop can be optimized accordingly.

While a loop may be more efficient in certain situations where the condition being tested can be evaluated quickly.

```
# 24
a=int(input("Enter the limit: "))
i=int(input("Enter 1 for odd 2 for even"))
while i<=a:
  print(i)
  i+=2
```

```
Enter the limit: 10
Enter 1 for odd 2 for even1
1
3
5
7
9
```

```
# 25
i=1
while i<8:
  print(i)
  if i==5:
    break
  i+=1
```

```
1
2
3
4
5
```

```
apple
banana
cherry
```

```
# 27

**The break statement** in Python is used to terminate the loop or
statement in which it is present. After that, the control will pass
statements that are present after the break statement, if available.
the break statement is present in the nested loop, then it terminate
those loops which contain the break statement.

**Continue** is also a loop control statement just like the break
statement. continue statement is opposite to that of the break state
instead of terminating the loop, it forces to execute the next itera
of the loop. As the name suggests the continue statement forces the
to continue or execute the next iteration. When the continue stateme
executed in the loop, the code inside the loop following the continu
statement will be skipped and the next iteration of the loop will be
```

27

The break statement in Python is used to terminate the loop or statement in which it is present. After that, the control will pass to the statements that are present after the break statement, if available. If the break statement is present in the nested loop, then it terminates only those loops which contain the break statement.

Continue is also a loop control statement just like the break statement. continue statement is opposite to that of the break statement, instead of terminating the loop, it forces to execute the next iteration of the loop. As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.

```
# 27
var_list=("apple","banana","cherry")
for i in var_list:
  print(i)
```

```
# 28
a = 'Sneha Bhadauria'
for i in range(len(a)):
  if i>4:
    break
  else:
    print(a[i])
```

```
S
n
e
h
a
```

```
# 29

def my_func(var_x):
  mylist=[]

  for x in range (0,var_x):
    mylist.append(input("Enter the argument: "))

  for x in range(0,var_x,2):
    print(mylist[x])

a=int(input("Enter number of arguments: "))
my_func(a);
```

```
Enter number of arguments: 5
Enter the argument: Karan
Enter the argument: Shaggy
Enter the argument: Sneha
Enter the argument: Rekha
Enter the argument: Shabby
Karan
Sneha
Shabby
```

```
# 30
```