Name: Akshat Kumar **Reg.No:** 20MIS0183

ASSESSMENT-5 (ACTIVITY 1.27)

1. List out the similarities and differences (in tabular column format) between the following in Python.

a. List
b. Dictionary
c. Tuples

List	Dictionary	Tuple
Data type in which values are separated by comma and placed inside square backets [ ]	**Data type in which values are associated with keys and placed inside curly brackets { }**	**Data type in which values are separated by comma and placed inside the brackets ( )**
We can access the values using their index number	**We can access the values using their key**	**We can access the values using their index number**
It is a mutable Data type, the values in the list can be altered	**It is also a mutable Data type as we could change the key value**	**It is immutable Data type, we cannot alter the values in the tuple**
It supports Iteration	**It also supports Iteration.**	**It also supports Iteration**
It supports negative indexing using which we could access values from last position	**It has no order like indexing, they can only be accessed using their keys**	**It supports negative indexing using which we could access values from last position**
Example my_list = [10, 20, 30, 40, 50]	**Example** my_dict = [“Name”: Akshat, “Age”:18]	**Example** my_tuple = (1, 2, 3, 4, 5)

2. Write a Python code using tuples to produce the following four outputs.

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
```

CODE:

```
tuple1=()
tuple2=(1, 2, 3)
tuple3=(1, "Hello", 3.4)
tuple4=("mouse", [8, 4, 6], (1, 2, 3))
print(tuple1)
print(tuple2)
print(tuple3)
print(tuple4)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\2.py"
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))

[Done] exited with code=0 in 0.481 seconds
```

3. Execute the following code and observe the output. Write the inference from the program and the output.

```
my_tuple = 3, 4.6, "God"

print(my_tuple)

a, b, c = my_tuple

print(a)

print(b)

print(c)
```

CODE:

```
my_tuple=3,4.6,"God"
print(my_tuple)
a,b,c=my_tuple
print(a)
print(b)
print(c)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\3.py"
(3, 4.6, 'God')
3
4.6
God

[Done] exited with code=0 in 1.989 seconds
```

INFERENCE:

At first a tuple named as **my_tuple** is created that contains 3 different values number, integer and a string, separated by comma. After that different variables are accessed by using multiple assignment. First variable is used to access the first element of the tuple, similarly second variable accesses the second element and so on.

4. Create an empty tuple (say tuple1). Using loop, read the name of some engineering colleges (at least 10) from users and feed them into the tuple1. Print the tuple1 after feeding each college name.

FOR LOOP

CODE:

```
college_list=[]
college_tuple=()
num=int(input("Enter the number of Engeering Colleges you want to add in Tuple : "))
for i in range(1,num+1):
    col=input("Enter the name of college {} : ".format(i))
    if col not in college_list:
        college_list.append(col)
        college_tuple=tuple(college_list)
        print(college_tuple)
```

OUTPUT:

```
C:\Users\user\Desktop\python>C:/Users/user/AppData/Local/Programs/Python/Python3
8-32/python.exe c:/Users/user/Desktop/python/act1.27/4_for.py

Enter the number of Engeering Colleges you want to add in Tuple : 10
Enter the name of college 1 : VIT
('VIT',)
Enter the name of college 2 : SRM
('VIT', 'SRM')
Enter the name of college 3 : RAMAIAH
('VIT', 'SRM', 'RAMAIAH')
Enter the name of college 4 : BIT MESRA
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA')
Enter the name of college 5 : JADHAVPUR
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR')
Enter the name of college 6 : LPU
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR', 'LPU')
Enter the name of college 7 : CHANDIGARH
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR', 'LPU', 'CHANDIGARH')
Enter the name of college 8 : BITS PILANI
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR', 'LPU', 'CHANDIGARH', 'BITS PILANI')
Enter the name of college 9 : IITKHARAGPUR
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR', 'LPU', 'CHANDIGARH', 'BITS PILANI', 'IITKHARAGPUR')
Enter the name of college 10 : NIT TRICHY
('VIT', 'SRM', 'RAMAIAH', 'BIT MESRA', 'JADHAVPUR', 'LPU', 'CHANDIGARH', 'BITS PILANI', 'IITKHARAGPUR', 'NIT
TRICHY')
```

4. Create an empty tuple (say tuple1). Using loop, read the name of some engineering colleges (at least 10) from users and feed them into the tuple1. Print the tuple1 after feeding each college name.

WHILE LOOP

CODE:

```
college_list=[]
college_tuple=()
col=int(input("Enter the number of colleges you want to tuple : "))
print("Enter name of College : ")
i=1
while i<=col:
    n=input("Name {} :".format(i))
    if n not in college_list:
        college_list.append(n)
    else:
        print("This is already entered !! Try another name")
        continue
    college_tuple=tuple(college_list)
    print(college_tuple)
    i+=1
```

OUTPUT:

```
C:\Users\user\Desktop\python>C:/Users/user/AppData/Local/Programs/Python/Python3
8-32/python.exe c:/Users/user/Desktop/python/act1.27/4_while.py

Enter the number of colleges you want to tuple : 10
Enter name of College :
Name 1 :VIT
('VIT',)
Name 2 :BITS PILANI
('VIT', 'BITS PILANI')
Name 3 :SAASTRA
('VIT', 'BITS PILANI', 'SAASTRA')
Name 4 :RAMAIAH
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH')
Name 5 :BM COLLEGE
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE')
Name 6 :IEM
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE', 'IEM')
Name 7 :SRM
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE', 'IEM', 'SRM')
Name 8 :LPU
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE', 'IEM', 'SRM', 'LPU')
Name 9 :SHARDA
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE', 'IEM', 'SRM', 'LPU', 'SHARDA')
Name 10 :KALINGA
('VIT', 'BITS PILANI', 'SAASTRA', 'RAMAIAH', 'BM COLLEGE', 'IEM', 'SRM', 'LPU', 'SHARDA', 'KALINGA')
```

5. Create an empty tuple (say tuple1). Using loop, feed the all even numbers that exist between 100 and 200 into the tuple1. Print the tuple1 after feeding even number.

FOR LOOP

CODE:

```
number_tuple=()
number_list=[]
for i in range(100,200):
    if i%2==0:
        number_list.append(i)
        number_tuple=tuple(number_list)
        print(number_tuple)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\5_for.py"

(100,)
(100, 102)
(100, 102, 104)
(100, 102, 104, 106)
(100, 102, 104, 106, 108)
(100, 102, 104, 106, 108, 110)
(100, 102, 104, 106, 108, 110, 112)
(100, 102, 104, 106, 108, 110, 112, 114)
(100, 102, 104, 106, 108, 110, 112, 114, 116)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154)
```

```
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194,
196)
(100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146,
148, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 188, 190, 192, 194,
196, 198)

[Done] exited with code=0 in 0.555 seconds
```

6. Write a python code to produce the following output using my_tuple = ('p','r','o','g','r','a','m','i','z')

```
('r', 'o', 'g')
('p', 'r')
('i', 'z')
('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
```

CODE:

```
my_tuple=('p','r','o','g','r','a','m','i','z')
print(my_tuple[1:4])
print(my_tuple[0:2])
print(my_tuple[-2:])
print(my_tuple)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\6.py"
('r', 'o', 'g')
('p', 'r')
('i', 'z')
('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

[Done] exited with code=0 in 0.506 seconds
```

7. Can we reassign tuples? Check this statement with the following codes. Show the corresponding outputs.

```
my_tuple = (4, 2, 3, [6, 5])
print(my_tuple)

my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

print(my_tuple)
```

CODE:

```
my_tuple=(4, 2, 3, [6, 5])
print(my_tuple)

my_tuple=('p','r','o','g','r','a','m','i','z')

print(my_tuple)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\7.py"
(4, 2, 3, [6, 5])
('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

[Done] exited with code=0 in 2.137 seconds
```

INFERENCE:

Yes we can reassign tuples.

From the above code we can see that we can reassign tuples with new set of values. As previously **my_tuple** value was (4, 2, 3, [6, 5]). After that **my_tuple** value was updated by (‘p’,‘r’,‘o’,‘g’,‘r’,‘a’,‘m’, ‘i’,‘z’).

8. Execute the following code and observe the output. Write the inference from the program and the output.

(a)

```
my_tuple = (4, 2, 3, [6, 5])
my_tuple[1] = 9
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\8a.py"
Traceback (most recent call last):
  File "c:\Users\user\Desktop\python\act1.27\8a.py", line 2, in <module>
    my_tuple[1]=9
TypeError: 'tuple' object does not support item assignment

[Done] exited with code=1 in 0.562 seconds
```

INFERENCE:

From the above output we get to know that :

Tuple is one of the 4 built-in data types in Python. As we know tuple is a collection which is ordered and **unchangeable**. So when we try to update the first index, the system shouts at us.

We cannot directly replace any data inside tuple.

8. Execute the following code and observe the output. Write the inference from the program and the output.

(b)

```
my_tuple = (4, 2, 3, [6, 5])
my_tuple[3][0] = 9
print(my_tuple)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\8b.py"
(4, 2, 3, [9, 5])

[Done] exited with code=0 in 0.485 seconds
```

INFERENCE:

From the above output we can observe that we have updated the value of item inside list, that is inside tuple. As we know tuple is immutable but, the item to be updated was inside nested list so it was changed. List items are changeable/ mutable.

8. Execute the following code and observe the output. Write the inference from the program and the output.

(c)

```
print((1, 2, 3) + (4, 5, 6))
('Repeat', 'Repeat', 'Repeat')
print(("India",) * 3)
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\8c.py"
(1, 2, 3, 4, 5, 6)
('India', 'India', 'India')

[Done] exited with code=0 in 0.611 seconds
```

INFERENCE:

From the above output we get to know that :

Here in this code the tuples are added using **“+”** opeartor. In output it gets **concatenated.**

And from the second output we get to know that we can print the value inside a tuple as many times as we want using **“*”** operator.

9. Execute the following code and observe the output.

```
my_tuple = ('a', 'p', 'p', 'l', 'e')

print(my_tuple.count('p'))

print(my_tuple.count('e'))

print(my_tuple.index('p'))

print(my_tuple.index('e'))
```

OUTPUT:

```
[Running] python -u "c:\Users\user\Desktop\python\act1.27\9.py"
2
1
1
4

[Done] exited with code=0 in 1.866 seconds
```

INFERENCE:

From the above output we get to know that :

count() function is used for counting the number of occurrences of a value in a tuple.

index() function returns the index of the given value in the tuple