

Python Lists & Tuples

Introduction to Lists

```
list1 = ["Rohan", "Physics", 21, 69.75]
list2 = [1, 2, 3, 4, 5]
list3 = ["a", "b", "c", "d"]
list4 = [25.50, True, -55, 1+2j]

# Display all lists
print("List 1:", list1)
print("List 2:", list2)
print("List 3:", list3)
print("List 4:", list4)
```

The screenshot shows the Online Python IDE interface. The code in main.py creates four lists: list1, list2, list3, and list4, and prints them. The output window displays the results:

```
List 1: ['Rohan', 'Physics', 21, 69.75]
List 2: [1, 2, 3, 4, 5]
List 3: ['a', 'b', 'c', 'd']
List 4: [25.5, True, -55, (1+2j)]
** Process exited - Return Code: 0 **
```

Accessing Values in Lists

```
# Accessing Values in Lists using Index

list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7]

print("list1[0]: ", list1[0])
print("list2[1:5]: ", list2[1:5])
```

The screenshot shows the Online Python IDE interface. The code in main.py prints elements from list1 and list2 using indexing. The output window displays the results:

```
list1[0]: physics
list2[1:5]: [2, 3, 4, 5]
** Process exited - Return Code: 0 **
```

Updating Lists

```
list = ['physics', 'chemistry', 1997, 2000];

print ("Value available at index 2 : ")

print (list[2])

list[2] = 2001;

print ("New value available at index 2 : ")

print (list[2])
```

The screenshot shows the Online Python interface. The code in main.py is:

```
list = ['physics', 'chemistry', 1997, 2000];
print ("Value available at index 2 : ")
print (list[2])
list[2] = 2001;
print ("New value available at index 2 : ")
print (list[2])
```

The output window shows the execution results:

```
Value available at index 2 :
1997
New value available at index 2 :
2001
** Process exited - Return Code: 0 **
```

Delete List Elements

```
# Delete List Elements

list1 = ['physics', 'chemistry', 1997, 2000];
print (list1)
del list1[2];
print ("After deleting value at index 2 : ")
print (list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
list1 = ['physics', 'chemistry', 1997, 2000];
print (list1)
del list1[2];
print ("After deleting value at index 2 : ")
print (list1)
```

The output window shows the execution results:

```
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
** Process exited - Return Code: 0 **
```

Basic List Operations

```
# Basic List Operations

len([1, 2, 3])           # Length
[1, 2, 3] + [4, 5, 6]    # Concatenation
['Hi!'] * 4               # Repetition
3 in [1, 2, 3]            # Membership

# Display results
print("Length:", len([1, 2, 3]))
print("Concatenation:", [1, 2, 3] + [4, 5, 6])
print("Repetition:", ['Hi!'] * 4)
print("Membership:", 3 in [1, 2, 3])
```

The screenshot shows the Online Python interface. The code in main.py is identical to the one above. The output pane displays the results of the print statements:

- Length: 3
- Concatenation: [1, 2, 3, 4, 5, 6]
- Repetition: ['Hi!', 'Hi!', 'Hi!', 'Hi!']
- Membership: True

At the bottom, it says: ** Process exited - Return Code: 0 **

Indexing, Slicing, and Matrixes

```
# Assuming following input

L = ['spam', 'Spam', 'SPAM!']

# Indexing Examples

print("L[2] =", L[2]) # Offsets start at zero

print("L[-2] =", L[-2] # Negative: count from the right

print("L[1:] =", L[1:])# Slicing fetches sections
```

The screenshot shows the Online Python interface. The code in main.py is identical to the one above. The output pane displays the results of the print statements:

- L[2] = SPAM!
- L[-2] = Spam
- L[1:] = ['Spam', 'SPAM!']

At the bottom, it says: ** Process exited - Return Code: 0 **

List Methods - append()

```
# List append() Method

list1 = ['physics', 'chemistry', 'maths']
list1.append('biology')
print("Updated list : ", list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List append() Method
2
3 list1 = ['physics', 'chemistry', 'maths']
4 list1.append('biology')
5 print("Updated list : ", list1)
6
```

The output window shows:

- Run button
- Share button
- Command Line Arguments button
- Output: Updated list : ['physics', 'chemistry', 'maths', 'biology']
- Process exited - Return Code: 0

List Methods - extend()

```
# List extend() Method

list1 = ['physics', 'chemistry', 'maths']
list2 = ['biology', 'history']
list1.extend(list2)
print("Extended list : ", list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List extend() Method
2
3 list1 = ['physics', 'chemistry', 'maths']
4 list2 = ['biology', 'history']
5 list1.extend(list2)
6 print("Extended list : ", list1)
7
```

The output window shows:

- Run button
- Share button
- Command Line Arguments button
- Output: Extended list : ['physics', 'chemistry', 'maths', 'biology', 'history']
- Process exited - Return Code: 0

List Methods - insert()

```
# List insert() Method

list1 = ['physics', 'chemistry', 'maths']
list1.insert(1, 'biology')
print("List after insertion : ", list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List insert() Method
2
3 list1 = ['physics', 'chemistry', 'maths']
4 list1.insert(1, 'biology')
5 print("List after insertion : ", list1)
6
```

The output window shows:

- Run button
- Share button
- Command Line Arguments button
- Output: List after insertion : ['physics', 'biology', 'chemistry', 'maths']
- Process exited - Return Code: 0

List Methods - remove()

```
# List remove() Method

list1 = ['physics', 'chemistry', 'maths', 'biology']
list1.remove('maths')
print("List after removal : ", list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List remove() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'biology']
4 list1.remove('maths')
5 print("List after removal : ", list1)
6
```

The output window shows:

```
List after removal : ['physics', 'chemistry', 'biology']
** Process exited - Return Code: 0 **
```

List Methods - pop()

```
# List pop() Method

list1 = ['physics', 'chemistry', 'maths', 'biology']
list1.pop()
print("List after pop() : ", list1)

list1.pop(1)
print("List after pop(1) : ", list1)
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List pop() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'biology']
4 list1.pop()
5 print("List after pop() : ", list1)
6
7 list1.pop(1)
8 print("List after pop(1) : ", list1)
9
```

The output window shows:

```
List after pop() : ['physics', 'chemistry', 'maths']
List after pop(1) : ['physics', 'maths']
** Process exited - Return Code: 0 **
```

List Methods - index()

```
# List index() Method

list1 = ['physics', 'chemistry', 'maths', 'biology']
print("Index of chemistry : ", list1.index('chemistry'))
```

The screenshot shows the Online Python interface. The code in main.py is:

```
1 # List index() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'biology']
4 print("Index of chemistry : ", list1.index('chemistry'))
5
```

The output window shows:

```
Index of chemistry : 1
** Process exited - Return Code: 0 **
```

List Methods - count()

```
# List count() Method

list1 = ['physics', 'chemistry', 'maths', 'chemistry']
print("Count of chemistry : ", list1.count('chemistry'))
```

ONLINE PYTHON

main.py

```
1 # List count() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'chemistry']
4 print("Count of chemistry : ", list1.count('chemistry'))
5
```

Run Share \$ Command Line Arguments

Count of chemistry : 2

** Process exited - Return Code: 0 **

List Methods - sort()

```
# List sort() Method

list1 = ['physics', 'chemistry', 'maths', 'biology']
list1.sort()
print("Sorted list : ", list1)
```

ONLINE PYTHON

main.py

```
1 # List sort() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'biology']
4 list1.sort()
5 print("Sorted list : ", list1)
6
```

Run Share \$ Command Line Arguments

Sorted list : ['biology', 'chemistry', 'maths', 'physics']

** Process exited - Return Code: 0 **

List Methods - reverse()

```
# List reverse() Method

list1 = ['physics', 'chemistry', 'maths', 'biology']
list1.reverse()
print("Reversed list : ", list1)
```

ONLINE PYTHON

main.py

```
1 # List reverse() Method
2
3 list1 = ['physics', 'chemistry', 'maths', 'biology']
4 list1.reverse()
5 print("Reversed list : ", list1)
6
```

Run Share \$ Command Line Arguments

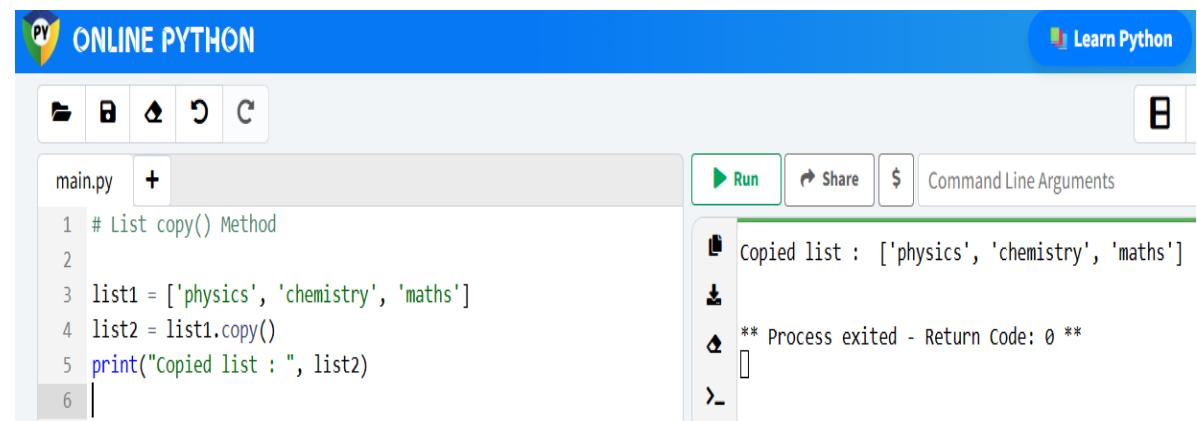
Reversed list : ['biology', 'maths', 'chemistry', 'physics']

** Process exited - Return Code: 0 **

List Methods - copy()

```
# List copy() Method

list1 = ['physics', 'chemistry', 'maths']
list2 = list1.copy()
print("Copied list : ", list2)
```

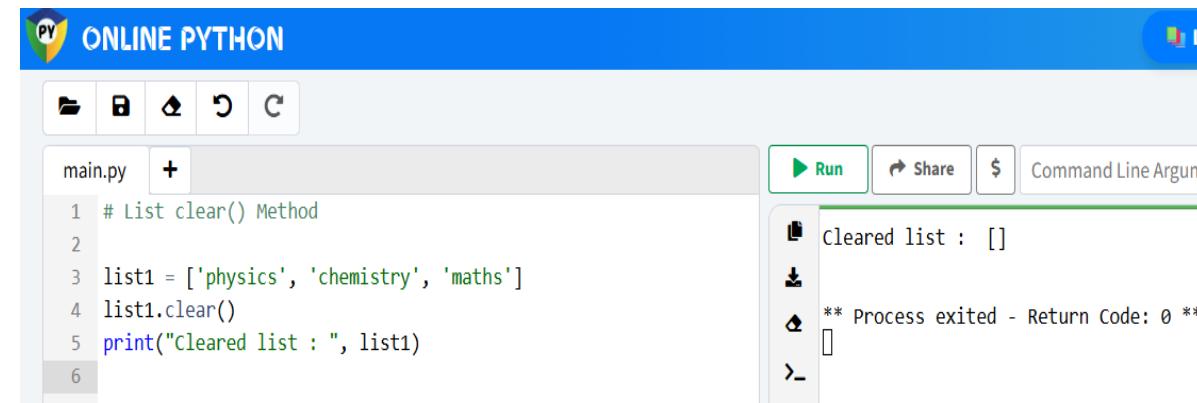


The screenshot shows the Online Python interface. In the code editor, there is a file named 'main.py' containing the provided Python code. The output window shows the result of running the code: 'Copied list : ['physics', 'chemistry', 'maths']'. Below this, it says '** Process exited - Return Code: 0 **'.

List Methods - clear()

```
# List clear() Method

list1 = ['physics', 'chemistry', 'maths']
list1.clear()
print("Cleared list : ", list1)
```



The screenshot shows the Online Python interface. In the code editor, there is a file named 'main.py' containing the provided Python code. The output window shows the result of running the code: 'Cleared list : []'. Below this, it says '** Process exited - Return Code: 0 **'.

Accessing List Items with Indexing

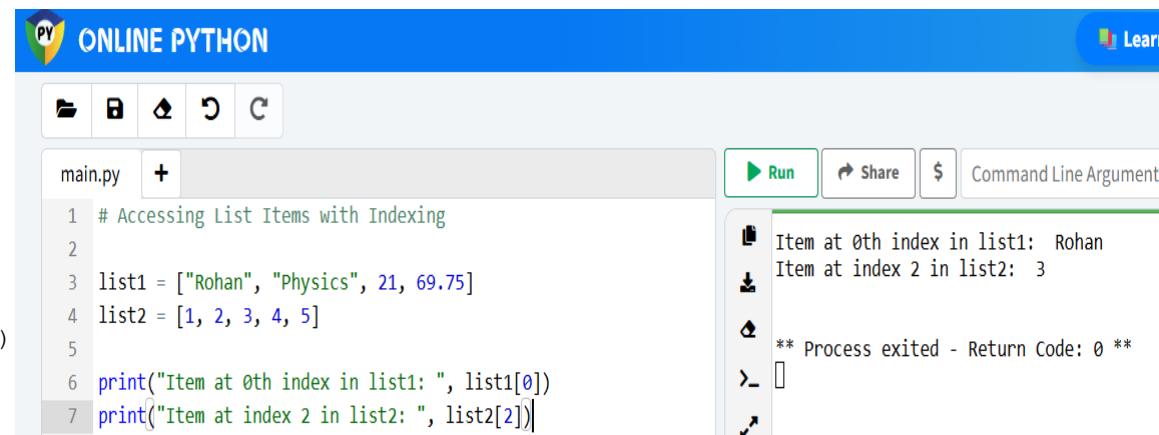
```
# Accessing List Items with Indexing

list1 = ["Rohan", "Physics", 21, 69.75]

list2 = [1, 2, 3, 4, 5]

print("Item at 0th index in list1: ", list1[0])

print("Item at index 2 in list2: ", list2[2])
```



The screenshot shows the Online Python IDE interface. The code in main.py is identical to the one above. The output window on the right shows the results of the print statements:

```
Item at 0th index in list1: Rohan
Item at index 2 in list2: 3
** Process exited - Return Code: 0 **
```

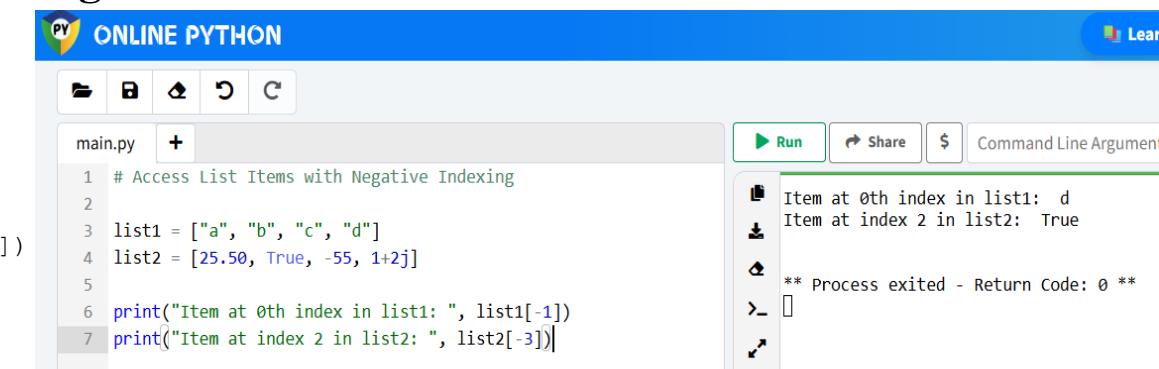
Access List Items with Negative Indexing

```
list1 = ["a", "b", "c", "d"]

list2 = [25.50, True, -55, 1+2j]

print("Item at 0th index in list1: ", list1[-1])

print("Item at index 2 in list2: ", list2[-3])
```



The screenshot shows the Online Python IDE interface. The code in main.py is identical to the one above. The output window on the right shows the results of the print statements:

```
Item at 0th index in list1: d
Item at index 2 in list2: True
** Process exited - Return Code: 0 **
```

Access List Items with Slice Operator

```
# Access List Items with Slice Operator

list1 = ["a", "b", "c", "d"]

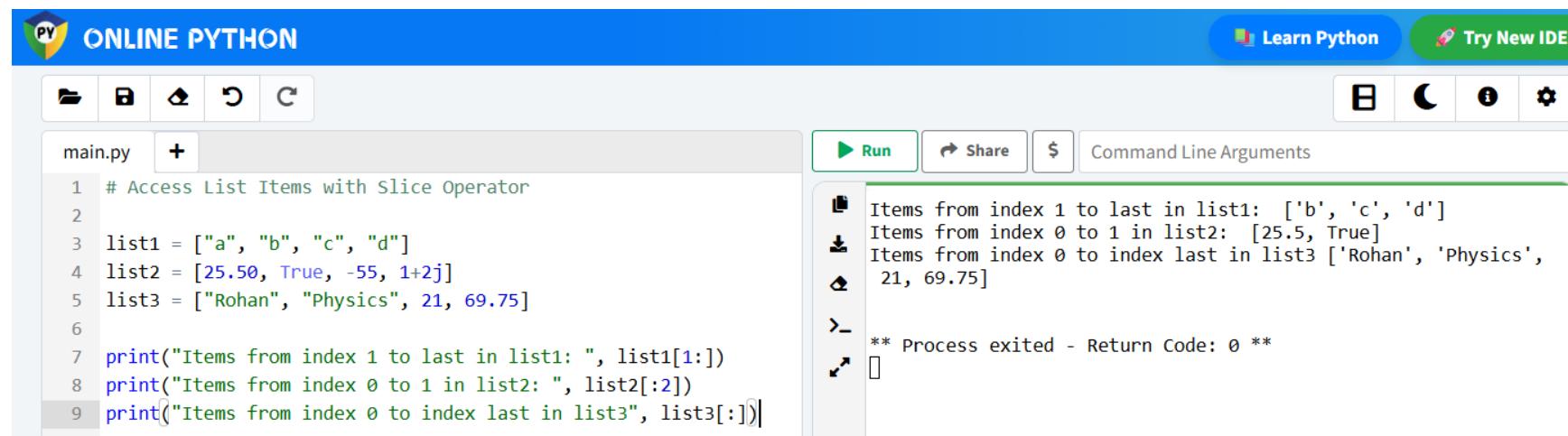
list2 = [25.50, True, -55, 1+2j]

list3 = ["Rohan", "Physics", 21, 69.75]

print("Items from index 1 to last in list1: ", list1[1:])

print("Items from index 0 to 1 in list2: ", list2[:2])

print("Items from index 0 to index last in list3", list3[:])
```



The screenshot shows the Online Python IDE interface. The code editor window contains the provided Python script. The output window displays the results of the print statements:

```
Items from index 1 to last in list1: ['b', 'c', 'd']
Items from index 0 to 1 in list2: [25.5, True]
Items from index 0 to index last in list3 ['Rohan', 'Physics', 21, 69.75]

** Process exited - Return Code: 0 **
```

Access Sub List from a List

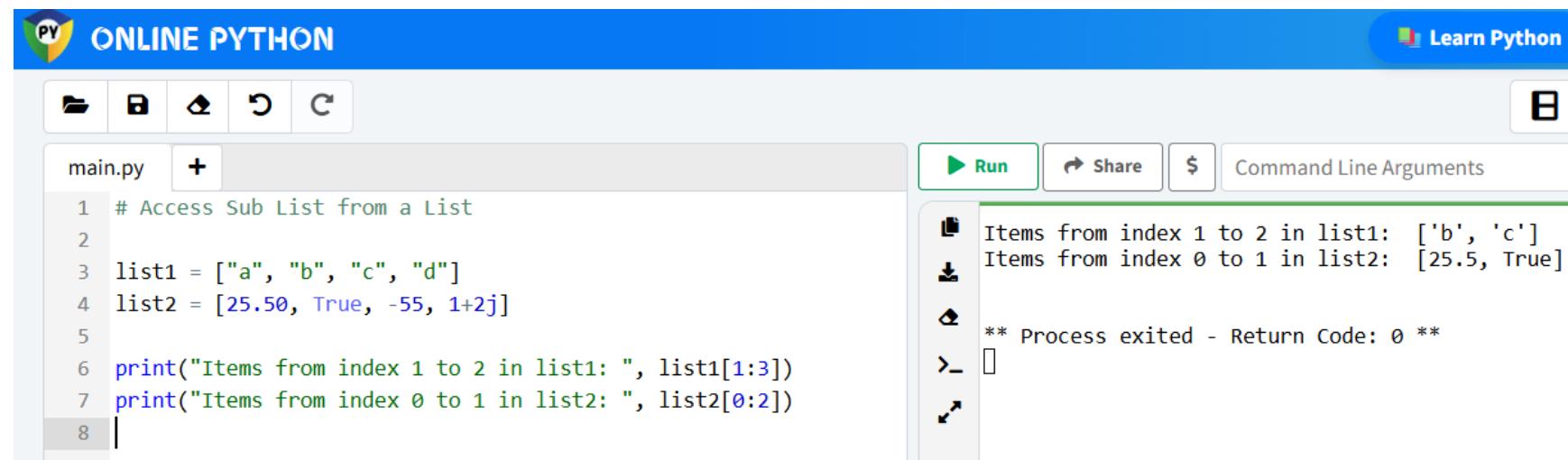
```
# Access Sub List from a List

list1 = ["a", "b", "c", "d"]

list2 = [25.50, True, -55, 1+2j]

print("Items from index 1 to 2 in list1: ", list1[1:3])

print("Items from index 0 to 1 in list2: ", list2[0:2])
```



The screenshot shows the Online Python IDE interface. The code editor window contains the provided Python script. The output window displays the results of the print statements:

```
Items from index 1 to 2 in list1: ['b', 'c']
Items from index 0 to 1 in list2: [25.5, True]
** Process exited - Return Code: 0 **
```

Python List Exercise 1

```
L1 = [1, 9, 1, 6, 3, 4, 5, 1, 2, 5, 6, 7, 8, 9, 2]
```

```
L2 = []
```

```
for x in L1:
```

```
    if x not in L2:
```

```
        L2.append(x)
```

```
print(L2)
```

ONLINE PYTHON

main.py

```
1 # Python program to find unique numbers in a given list
2
3 L1 = [1, 9, 1, 6, 3, 4, 5, 1, 2, 5, 6, 7, 8, 9, 2]
4 L2 = []
5
6 for x in L1:
7     if x not in L2:
8         L2.append(x)
9
10 print(L2)
```

[1, 9, 6, 3, 4, 5, 2, 7, 8]

** Process exited - Return Code: 0 **

Python List Exercise 2

```
L1 = [1, 9, 1, 6, 3, 4]
```

```
ttl = 0
```

```
for x in L1:
```

```
    ttl += x
```

```
print("Sum of all numbers Using loop:", ttl)
```

```
ttl = sum(L1)
```

```
print("Sum of all numbers using sum() function:", ttl)
```

ONLINE PYTHON

main.py

```
1 # Python program to find sum of all numbers in a list
2
3 L1 = [1, 9, 1, 6, 3, 4]
4 ttl = 0
5
6 for x in L1:
7     ttl += x
8
9 print("Sum of all numbers Using loop:", ttl)
10
11 ttl = sum(L1)
12 print("Sum of all numbers using sum() function:", ttl)
13
```

Sum of all numbers Using loop: 24

Sum of all numbers using sum() function: 24

** Process exited - Return Code: 0 **

Python List Exercise 3

```
import random

L1 = []

for i in range(5):
    x = random.randint(0, 100)
    L1.append(x)

print (L1)
```

ONLINE PYTHON

The screenshot shows the 'ONLINE PYTHON' interface. On the left, there's a code editor window titled 'main.py' containing the provided Python script. To the right of the code editor is a terminal window showing the output of the script's execution. The terminal output includes the list [38, 99, 62, 70, 69] and the message '** Process exited - Return Code: 0 **'. At the top of the interface, there are several icons: a folder, a file, a download arrow, a refresh arrow, a copy/cut icon, and a paste icon. To the right of these icons are buttons for 'Run' (highlighted in green), 'Share', and 'Command Line Arguments'. The overall layout is clean and functional, typical of an online coding environment.

```
main.py +  
1 import random  
2 L1 = []  
3 for i in range(5):  
4     x = random.randint(0, 100)  
5     L1.append(x)  
6 print (L1)  
[38, 99, 62, 70, 69]  
** Process exited - Return Code: 0 **
```

Introduction to Tuples

```
# Examples of Python Tuples

tup1 = ("Rohan", "Physics", 21, 69.75)

tup2 = (1, 2, 3, 4, 5)

tup3 = ("a", "b", "c", "d")

tup4 = (25.50, True, -55, 1+2j)

# Empty tuple

tup_empty = ()

# Tuple with a single value (note the comma)

tup_single = (50,)

print("tup1:", tup1)

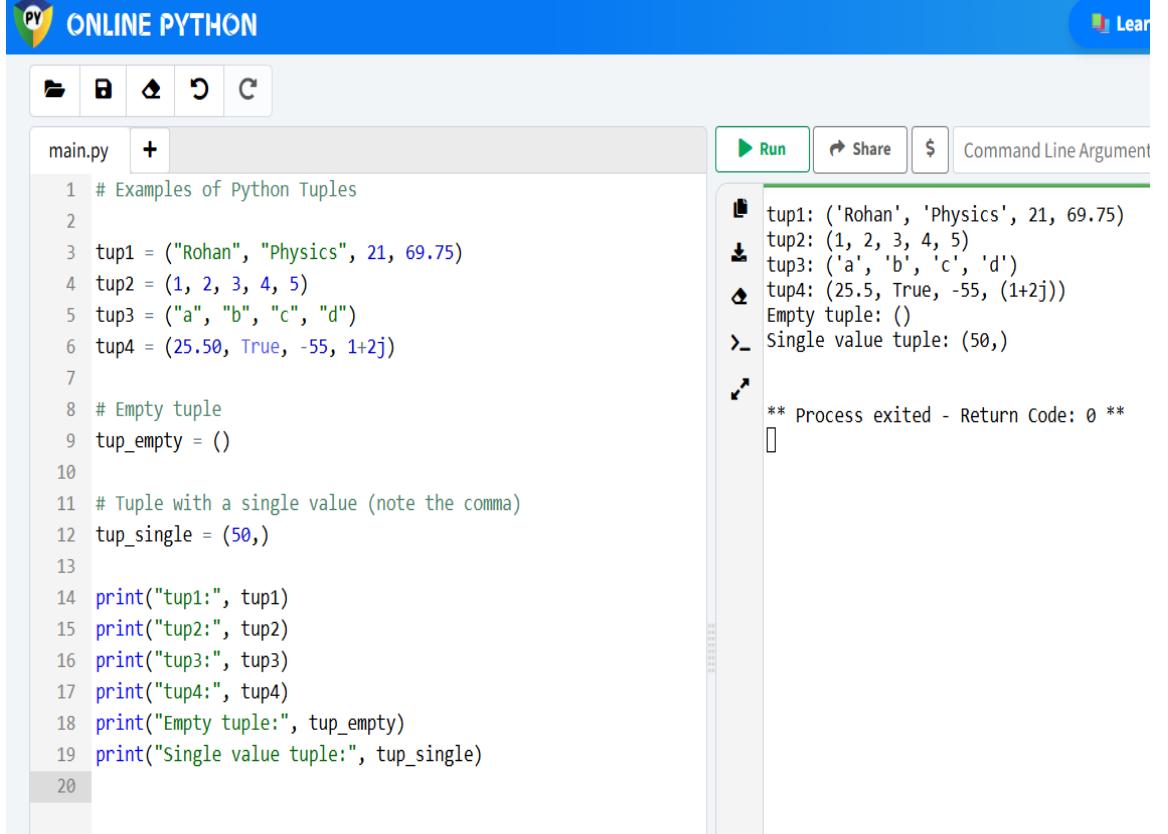
print("tup2:", tup2)

print("tup3:", tup3)

print("tup4:", tup4)

print("Empty tuple:", tup_empty)

print("Single value tuple:", tup_single)
```



The screenshot shows the Online Python IDE interface. The code editor window contains the provided Python script. The output window on the right displays the results of each print statement:

```
tup1: ('Rohan', 'Physics', 21, 69.75)
tup2: (1, 2, 3, 4, 5)
tup3: ('a', 'b', 'c', 'd')
tup4: (25.5, True, -55, (1+2j))
Empty tuple: ()
Single value tuple: (50,)

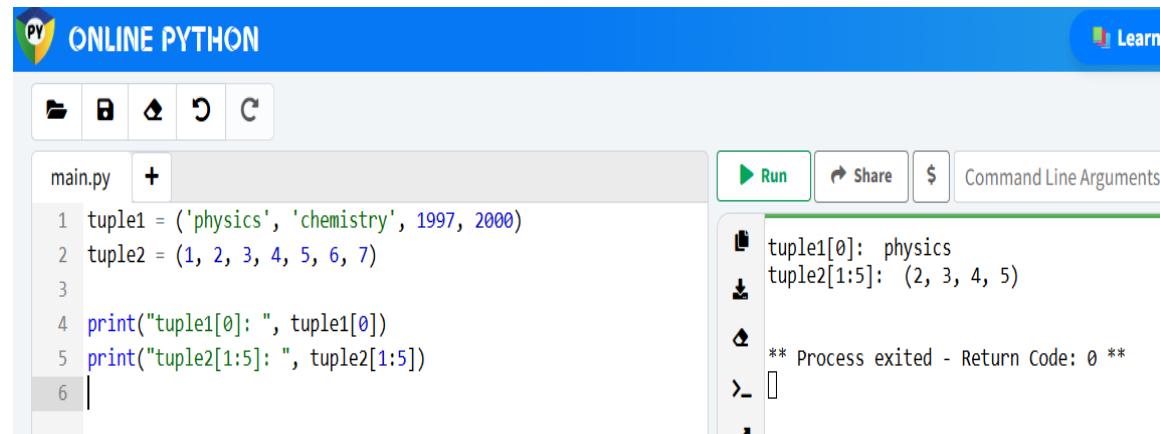
** Process exited - Return Code: 0 **
```

Accessing Values in Tuples

```
# Accessing Values in Tuples

tuple1 = ('physics', 'chemistry', 1997, 2000)
tuple2 = (1, 2, 3, 4, 5, 6, 7)

print("tuple1[0]: ", tuple1[0])
print("tuple2[1:5]: ", tuple2[1:5])
```



The screenshot shows the Online Python IDE interface. The code in main.py is:

```
1 tuple1 = ('physics', 'chemistry', 1997, 2000)
2 tuple2 = (1, 2, 3, 4, 5, 6, 7)
3
4 print("tuple1[0]: ", tuple1[0])
5 print("tuple2[1:5]: ", tuple2[1:5])
6
```

The output window shows the results of the print statements:

```
tuple1[0]: physics
tuple2[1:5]: (2, 3, 4, 5)
** Process exited - Return Code: 0 **
```

Updating Tuples (Immutable)

```
# Tuples are immutable, meaning you cannot update or change their elements

tup1 = (12, 34.56)

tup2 = ('abc', 'xyz')

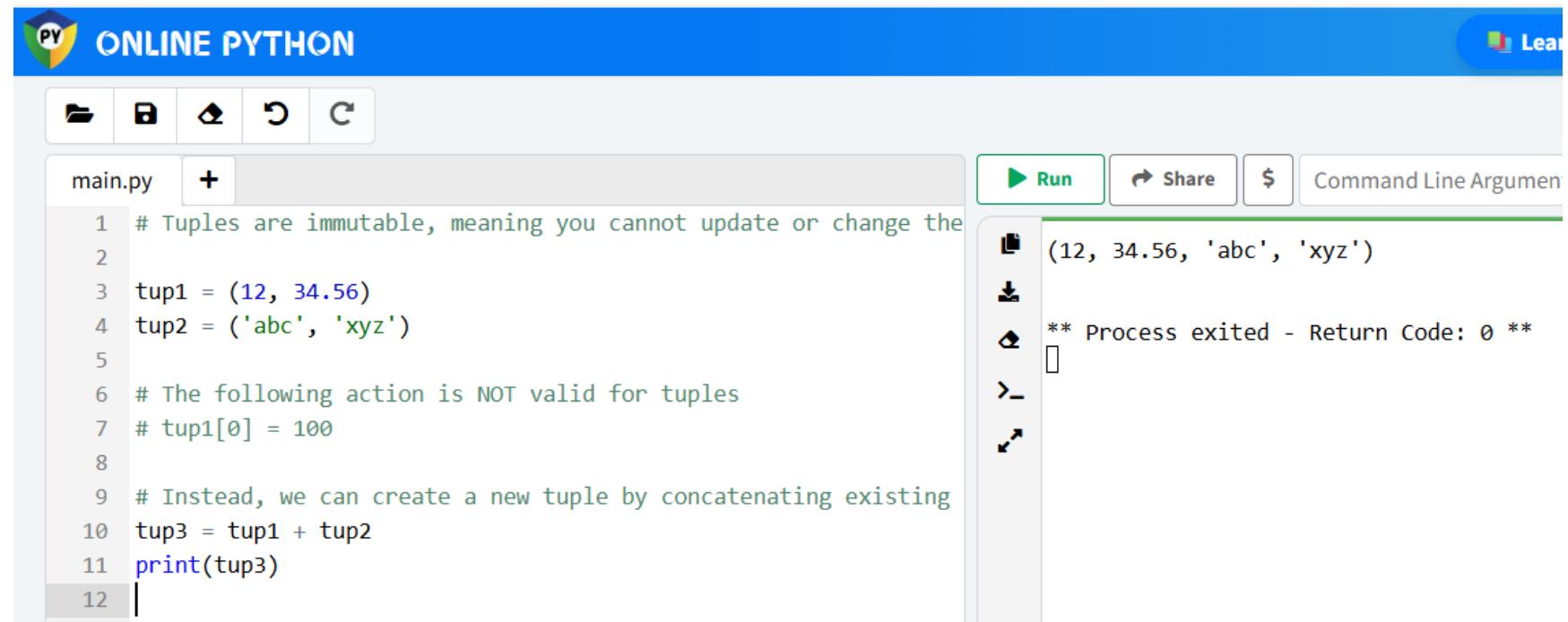
# The following action is NOT valid for tuples

# tup1[0] = 100

# Instead, we can create a new tuple by concatenating existing ones

tup3 = tup1 + tup2

print(tup3)
```



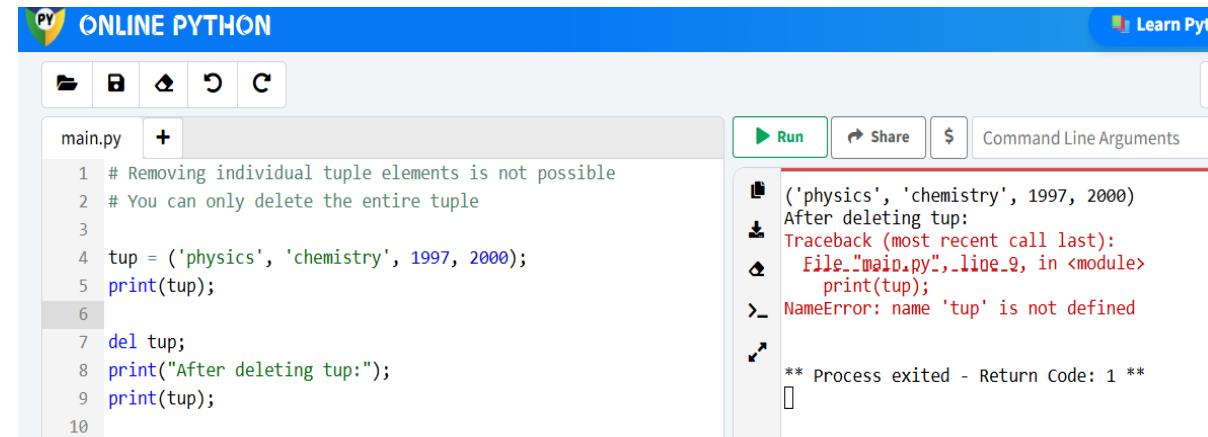
The screenshot shows the Online Python IDE interface. The top bar has a logo and the text "ONLINE PYTHON". Below the bar are file management icons: a folder, a file, a download, a refresh, and a search. The main area contains a code editor with a file named "main.py" containing Python code. The code demonstrates tuple immutability and tuple concatenation. The output panel on the right shows the result of running the code, which is a tuple "(12, 34.56, 'abc', 'xyz')". It also displays the message "Process exited - Return Code: 0".

```
main.py +  
1 # Tuples are immutable, meaning you cannot update or change the  
2  
3 tup1 = (12, 34.56)  
4 tup2 = ('abc', 'xyz')  
5  
6 # The following action is NOT valid for tuples  
7 # tup1[0] = 100  
8  
9 # Instead, we can create a new tuple by concatenating existing  
10 tup3 = tup1 + tup2  
11 print(tup3)  
12 |
```

(12, 34.56, 'abc', 'xyz')
** Process exited - Return Code: 0 **

Delete Tuple Elements

```
# Removing individual tuple elements is not possible  
  
# You can only delete the entire tuple  
  
tup = ('physics', 'chemistry', 1997, 2000)  
print(tup)  
  
del tup  
print("After deleting tup:")  
print(tup);
```



The screenshot shows the Online Python IDE interface. The top bar has a logo and the text "ONLINE PYTHON". Below the bar are file management icons: a folder, a file, a download, a refresh, and a search. The main area contains a code editor with a file named "main.py" containing Python code. The code attempts to delete individual tuple elements and prints the tuple after deletion. The output panel on the right shows an error message: "NameError: name 'tup' is not defined". It also displays the message "Process exited - Return Code: 1".

```
main.py +  
1 # Removing individual tuple elements is not possible  
2 # You can only delete the entire tuple  
3  
4 tup = ('physics', 'chemistry', 1997, 2000);  
5 print(tup);  
6  
7 del tup;  
8 print("After deleting tup:")  
9 print(tup);  
10 |
```

('physics', 'chemistry', 1997, 2000)
After deleting tup:
Traceback (most recent call last):
File "main.py", line 9, in <module>
 print(tup);
NameError: name 'tup' is not defined

** Process exited - Return Code: 1 **

Basic Tuple Operations

```
# --- Python Tuple Operations ---

# Concatenation

print("Tuple Concatenation:")
print((1, 2, 3) + (4, 5, 6))    # joins two tuples
print()

# Repetition

print("Tuple Repetition:")
print('Hi!',) * 4    # repeats the tuple 4 times
print()

# Membership

print("Tuple Membership:")
print(3 in (1, 2, 3))    # checks if 3 is present in the tuple
print()

# --- Indexing, Slicing, and Matrices ---

L = ('spam', 'Spam', 'SPAM!')

print("Indexing and Slicing Examples:")

print("L[2] =", L[2])      # element at index 2

print("L[-2] =", L[-2])    # element from the end (2nd last)

print("L[1:] =", L[1:])    # slice from index 1 to the end
```

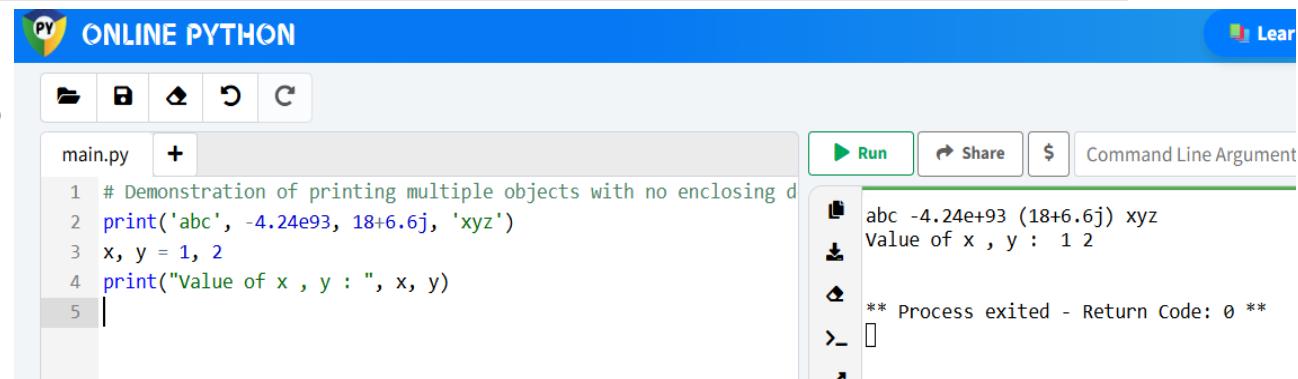
The screenshot shows the 'ONLINE PYTHON' interface. The code in 'main.py' is identical to the one above. The right panel displays the output of the print statements:

- Tuple Concatenation: (1, 2, 3, 4, 5, 6)
- Tuple Repetition: ('Hi!', 'Hi!', 'Hi!', 'Hi!')
- Tuple Membership: True
- Indexing and Slicing Examples:
 - L[2] = SPAM!
 - L[-2] = Spam
 - L[1:] = ('Spam', 'SPAM!')

At the bottom, it says: ** Process exited - Return Code: 0 **

No Enclosing Delimiters

```
print('abc', -4.24e93, 18+6.6j, 'xyz')  
  
x, y = 1, 2  
  
print("Value of x , y : ", x, y)
```



The screenshot shows the Online Python interface. The code in main.py is:

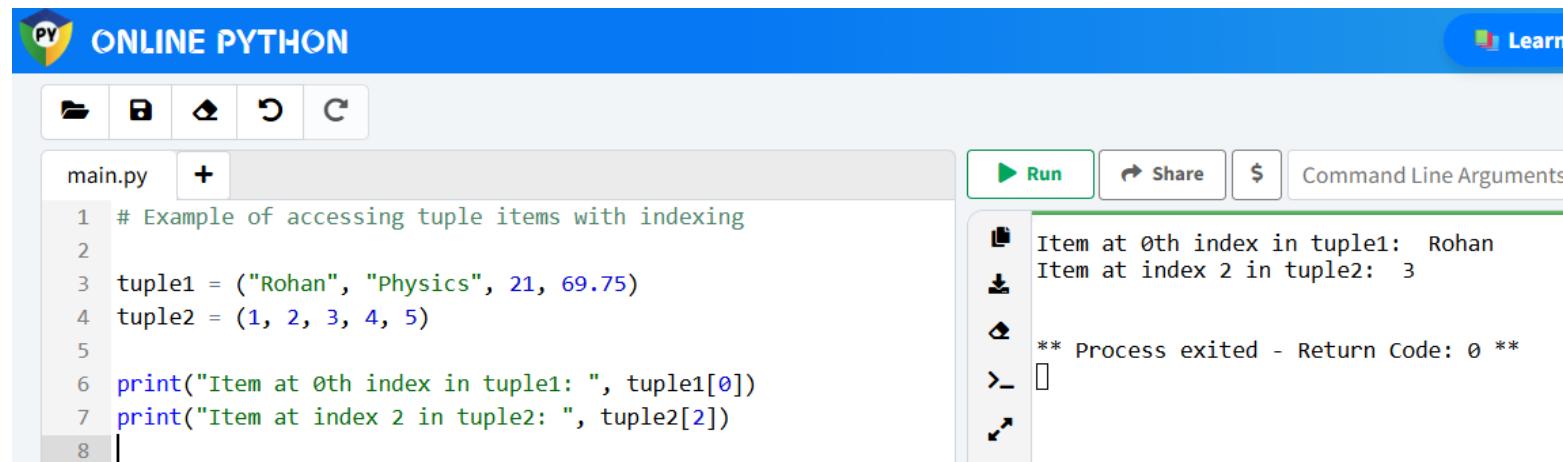
```
1 # Demonstration of printing multiple objects with no enclosing delimiters  
2 print('abc', -4.24e93, 18+6.6j, 'xyz')  
3 x, y = 1, 2  
4 print("Value of x , y : ", x, y)  
5
```

The output window shows the results of the print statements:

```
abc -4.24e+93 (18+6.6j) xyz  
Value of x , y : 1 2  
** Process exited - Return Code: 0 **
```

Accessing Tuple Items with Indexing

```
tuple1 = ("Rohan", "Physics", 21, 69.75)  
  
tuple2 = (1, 2, 3, 4, 5)  
  
print("Item at 0th index in tuple1: ", tuple1[0])  
  
print("Item at index 2 in tuple2: ", tuple2[2])
```



The screenshot shows the Online Python interface. The code in main.py is:

```
1 # Example of accessing tuple items with indexing  
2  
3 tuple1 = ("Rohan", "Physics", 21, 69.75)  
4 tuple2 = (1, 2, 3, 4, 5)  
5  
6 print("Item at 0th index in tuple1: ", tuple1[0])  
7 print("Item at index 2 in tuple2: ", tuple2[2])  
8
```

The output window shows the results of the print statements:

```
Item at 0th index in tuple1: Rohan  
Item at index 2 in tuple2: 3  
** Process exited - Return Code: 0 **
```

Accessing Tuple Items with Negative Indexing

```
# Accessing Tuple Items with Negative Indexing

tup1 = ("a", "b", "c", "d")

tup2 = (25.50, True, -55, 1+2j)

print("Item at 0th index in tup1: ", tup1[-1])

print("Item at index 2 in tup2: ", tup2[-3])
```

The screenshot shows the Online Python interface. The code in main.py is identical to the one above. The output window shows the results of the print statements:

```
Item at 0th index in tup1: d
Item at index 2 in tup2: True
** Process exited - Return Code: 0 **
```

Accessing Range of Tuple Items with Negative Indexing

```
# Accessing Range of Tuple Items with Negative Indexing

tup1 = ("a", "b", "c", "d")
tup2 = (1, 2, 3, 4, 5)

print("Items from index 1 to last in tup1: ", tup1[1:])
print("Items from index 2 to last in tup2: ", tup2[2:-1])
```

The screenshot shows the Online Python interface. The code in main.py is identical to the one above. The output window shows the results of the print statements:

```
Items from index 1 to last in tup1: ('b', 'c', 'd')
Items from index 2 to last in tup2: (3, 4)
** Process exited - Return Code: 0 **
```

Access Tuple Items with Slice Operator

```
# Access Tuple Items with Slice Operator
```

```
tuple1 = ("a", "b", "c", "d")
tuple2 = (25.50, True, -55, 1+2j)
tuple3 = (1, 2, 3, 4, 5)
tuple4 = ("Rohan", "Physics", 21, 69.75)

print("Items from index 1 to last in tuple1: ", tuple1[1:])
print("Items from index 0 to 1 in tuple2: ", tuple2[:2])
print("Items from index 0 to last in tuple3: ", tuple3[:])
```

The screenshot shows the Online Python IDE interface. The top bar includes the logo, 'ONLINE PYTHON', 'Learn Python', and 'Try' buttons. Below the bar are file management icons (New, Open, Save, Copy, Paste). The main area has tabs for 'main.py' and '+'. The code editor contains the provided Python script. To the right is a terminal window showing the output of the print statements. The terminal also displays a message about the process exiting.

```
1 # Access Tuple Items with Slice Operator
2
3 tuple1 = ("a", "b", "c", "d")
4 tuple2 = (25.50, True, -55, 1+2j)
5 tuple3 = (1, 2, 3, 4, 5)
6 tuple4 = ("Rohan", "Physics", 21, 69.75)
7
8 print("Items from index 1 to last in tuple1: ", tuple1[1:])
9 print("Items from index 0 to 1 in tuple2: ", tuple2[:2])
10 print("Items from index 0 to last in tuple3: ", tuple3[:])
11 |
```

```
Run Share $ Command Line Arguments
```

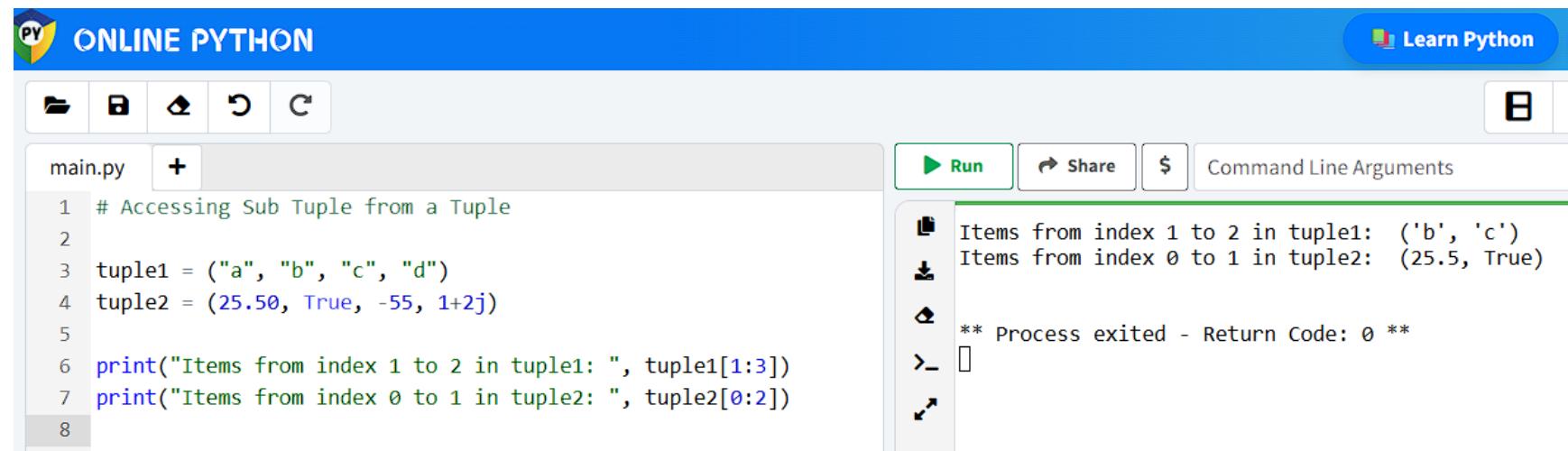
```
Items from index 1 to last in tuple1: ('b', 'c', 'd')
Items from index 0 to 1 in tuple2: (25.5, True)
Items from index 0 to last in tuple3: (1, 2, 3, 4, 5)
** Process exited - Return Code: 0 **
```

Accessing Sub Tuple from a Tuple

```
# Accessing Sub Tuple from a Tuple
```

```
tuple1 = ("a", "b", "c", "d")
tuple2 = (25.50, True, -55, 1+2j)

print("Items from index 1 to 2 in tuple1: ", tuple1[1:3])
print("Items from index 0 to 1 in tuple2: ", tuple2[0:2])
```



The screenshot shows the Online Python IDE interface. The code editor window contains a file named 'main.py' with the provided Python code. The output window shows the results of running the code, including the printed tuples and a success message.

```
ONLINE PYTHON Learn Python
```

File | Run | Share | \$ | Command Line Arguments

main.py +

```
1 # Accessing Sub Tuple from a Tuple
2
3 tuple1 = ("a", "b", "c", "d")
4 tuple2 = (25.50, True, -55, 1+2j)
5
6 print("Items from index 1 to 2 in tuple1: ", tuple1[1:3])
7 print("Items from index 0 to 1 in tuple2: ", tuple2[0:2])
8
```

Run Output:

```
Items from index 1 to 2 in tuple1: ('b', 'c')
Items from index 0 to 1 in tuple2: (25.5, True)
** Process exited - Return Code: 0 **
```

Python Tuple Exercise 1

```
T1 = (1, 9, 1, 6, 3, 4, 5, 1, 1, 2, 5, 6, 7, 8, 9, 2)
T2 = ()

for x in T1:
    if x not in T2:
        T2 += (x,)

print("Original tuple:", T1)
print("Unique numbers:", T2)
```



The screenshot shows the Online Python IDE interface. The code editor window contains the provided Python script. The output window displays the results of running the code, including the original tuple and the unique numbers found.

```
ONLINE PYTHON
Learn Python
Run Share $ Command Line Arguments

main.py +
1 # Python Tuple Exercise 1
2 # Program to find unique numbers in a given tuple
3
4 T1 = (1, 9, 1, 6, 3, 4, 5, 1, 1, 2, 5, 6, 7, 8, 9, 2)
5 T2 = ()
6
7 for x in T1:
8     if x not in T2:
9         T2 += (x,)
10
11 print("Original tuple:", T1)
12 print("Unique numbers:", T2)
13 |
```

```
Original tuple: (1, 9, 1, 6, 3, 4, 5, 1, 1, 2, 5, 6, 7, 8, 9, 2)
Unique numbers: (1, 9, 6, 3, 4, 5, 2, 7, 8)
** Process exited - Return Code: 0 **
```

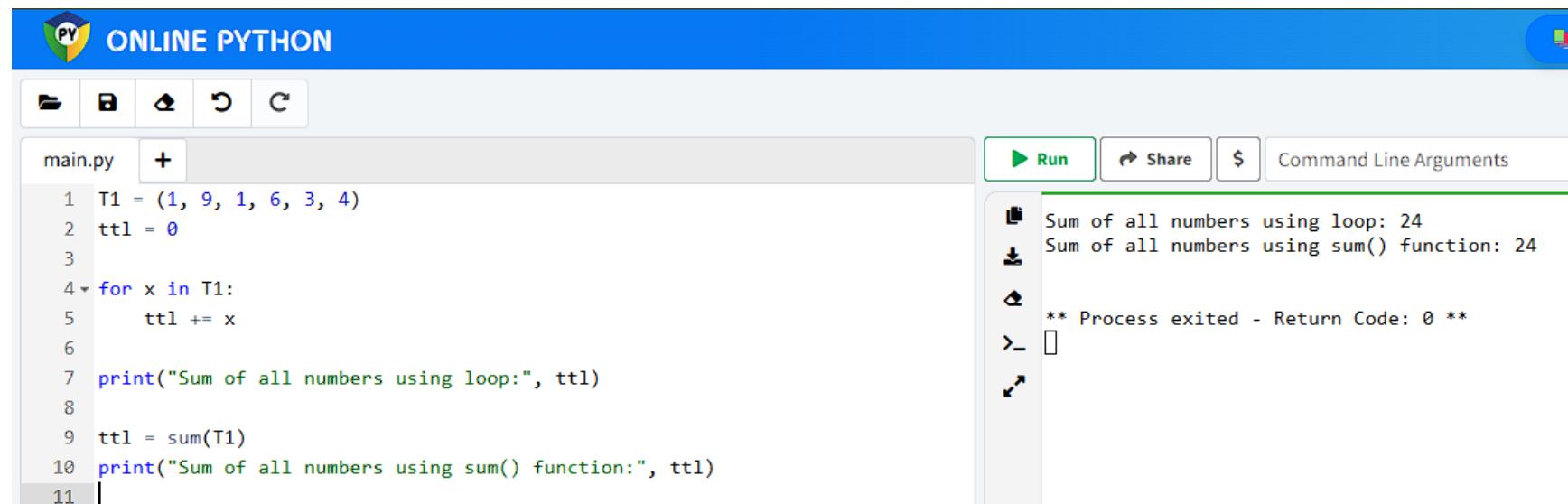
Python Tuple Exercise 2

```
T1 = (1, 9, 1, 6, 3, 4)
ttl = 0

for x in T1:
    ttl += x

print("Sum of all numbers using loop:", ttl)

ttl = sum(T1)
print("Sum of all numbers using sum() function:", ttl)
```



The screenshot shows the Online Python IDE interface. The code editor window contains the provided Python script. The output window displays the results of running the code, showing two identical outputs: "Sum of all numbers using loop: 24" and "Sum of all numbers using sum() function: 24". It also indicates that the process exited successfully with a return code of 0.

```
main.py + Run Share $ Command Line Arguments
1 T1 = (1, 9, 1, 6, 3, 4)
2 ttl = 0
3
4 for x in T1:
5     ttl += x
6
7 print("Sum of all numbers using loop:", ttl)
8
9 ttl = sum(T1)
10 print("Sum of all numbers using sum() function:", ttl)
11 |
```

Sum of all numbers using loop: 24
Sum of all numbers using sum() function: 24
** Process exited - Return Code: 0 **

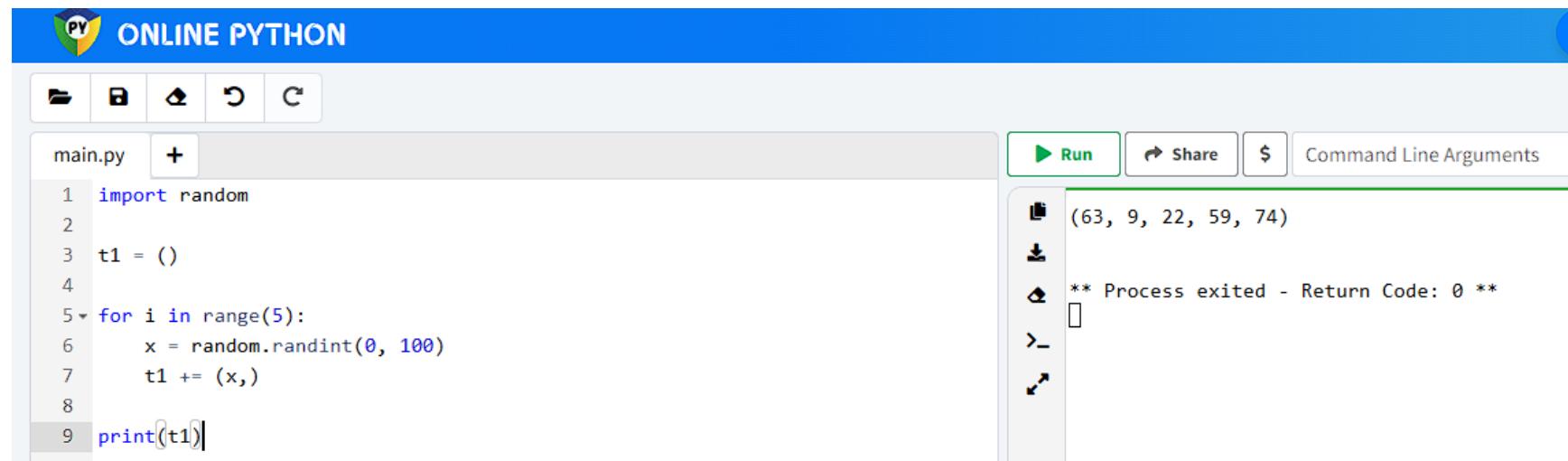
Python Tuple Exercise 3

```
import random

t1 = ()

for i in range(5):
    x = random.randint(0, 100)
    t1 += (x,)

print(t1)
```



The screenshot shows the Online Python interface. The code editor window contains the provided Python script. The output window shows the result of running the code, which is a tuple of five random integers: (63, 9, 22, 59, 74). Below the output, a message indicates the process exited successfully with a return code of 0.

```
ONLINE PYTHON
```

```
main.py +
```

```
1 import random
2
3 t1 = ()
4
5 for i in range(5):
6     x = random.randint(0, 100)
7     t1 += (x,)
8
9 print(t1)
```

```
▶ Run Share $ Command Line Arguments
```

```
(63, 9, 22, 59, 74)
** Process exited - Return Code: 0 **
```

Python NamedTuple Example: Employee Information

```
from collections import namedtuple

Employee = namedtuple("Employee", ["name", "age", "country"])

def main():
    try:
        employees = [
            Employee("Klaes Susana", 35, "USA"),
            Employee("Auxentius Cloe", 44, "Canada"),
            Employee("Golzar Merob", 28, "UK"),
            Employee("Tatjana Adhelm", 30, "Australia"),
        ]
        for employee in employees:
            print("Employee Name:", employee.name)
            print("Employee Country:", employee.country)
            print() # Empty line for separation

    except Exception as e:
        print("An error occurred:", e)

if __name__ == "__main__":
    main()
```

ONLINE PYTHON

main.py +

```
1 # Python NamedTuple Example: Employee Information
2
3 from collections import namedtuple
4
5 # Define the NamedTuple Employee
6 Employee = namedtuple("Employee", ["name", "age", "country"])
7
8 def main():
9     try:
10         # Create a list of Employee instances
11         employees = [
12             Employee("Klaes Susana", 35, "USA"),
13             Employee("Auxentius Cloe", 44, "Canada"),
14             Employee("Golzar Merob", 28, "UK"),
15             Employee("Tatjana Adhelm", 30, "Australia"),
16         ]
17
18         # Print each employee's name and country
19         for employee in employees:
20             print("Employee Name:", employee.name)
21             print("Employee Country:", employee.country)
22             print() # Empty line for separation
23
24     except Exception as e:
25         print("An error occurred:", e)
26
27 if __name__ == "__main__":
28     main()
29
```

Run Share \$ Command Line Argument

Employee Name: Klaes Susana
Employee Country: USA

Employee Name: Auxentius Cloe
Employee Country: Canada

Employee Name: Golzar Merob
Employee Country: UK

Employee Name: Tatjana Adhelm
Employee Country: Australia

** Process exited - Return Code: 0 **