1

```
#integer declaration

a = 43
a
```

```
43
```

2

```
#long variable

b = 1002345890372 # decimal
b_oct = 16454036101104
b_hex = E960788244

```

3

```
c = 4.2
c
```

```
4.2
```

4

```
x,y = 6,5

d_complex = complex(x,y)

print(d_complex.real, "+", d_complex.imag, "i")
```

```
6.0 + 5.0 i
```

5

```python
a_int, a_long, a_float, a_string = 10, 1234567890, 3.14, "Hello"
print(a_int,"\n",a_long,"\n", a_float,"\n", a_string)
```

```
10
 1234567890
 3.14
 Hello
```

6

```python
e_float = 4.2
print(e_float)
del e_float

```

```
4.2
```

7

!/usr/bin/python

is written at the beginning of a Python script to specify the path to the Python interpreter that should be used to execute the script. It's called a "shebang" or "hashbang". This line tells the shell which interpreter to use to run the script. In this case, it specifies that the Python interpreter located at /usr/bin/python should be used. This is particularly useful when you have multiple versions of Python installed on your system and want to specify which one to use for a particular script.

8. Declare a string variable and perform the following task.

a. Print complete string.

b. Print the first character of the string.

c. Print the characters starting from 3rd to 5th.

d. Print the string starting with the 3rd character.

e. Print the string two times.

f. Concatenated string

```
l_string = "Hi Prateek!"

# a. Print complete string.
print(l_string)

# b. Print the first character of the string.
print(l_string[0])

# c. Print the characters starting from 3rd to 5th.
print(l_string[2:5])

# d. Print the string starting with the 3rd character.
print(l_string[2:])

# e. Print the string two times.
print(l_string * 2)

# f. Concatenated string.
l_new_string = " How are you?"
print(l_string + l_new_string)

```

```
Hi Prateek!
H
 Pr
 Prateek!
Hi Prateek!Hi Prateek!
Hi Prateek! How are you?
```

9. Declare a list variable and perform the following task.

a. Print the complete list.

b. Print the first element of the list.

c. Print the elements list starting from 2nd till 3rd.

d. Print the elements list starting from the 3rd element.

e. Print the list two times.

f. Print the concatenated lists.

```python
m_list = [1, 2, 3, 4, 5]

# a. Print the complete list.
print(m_list)

# b. Print the first element of the list.
print(m_list[0])

# c. Print the elements list starting from 2nd till 3rd.
print(m_list[1:3])

# d. Print the elements list starting from the 3rd element.
print(m_list[2:])

# e. Print the list two times.
print(m_list * 2)

# f. Print the concatenated lists.
m_new_list = [6, 7, 8]
print(m_list + m_new_list)

```

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

10. Declare a tuple variable and perform the following task.

a. Print the complete tuple.

b. Print the first element of the tuple.

c. Print the elements tuple starting from 2nd till 3rd.

d. Print the elements tuple starting from 3rd element.

e. Print the tuple two times.

f. Print the concatenated tuple.

```python
n_tuple = (1, 2, 3, 4, 5)

# a. Print the complete tuple.
print(n_tuple)

# b. Print the first element of the tuple.
print(n_tuple[0])

# c. Print the elements tuple starting from 2nd till 3rd.
print(n_tuple[1:3])

# d. Print the elements tuple starting from 3rd element.
print(n_tuple[2:])

# e. Print the tuple two times.
print(n_tuple * 2)

# f. Print the concatenated tuple.
n_new_tuple = (6, 7, 8)
print(n_tuple + n_new_tuple)

```

```
(1, 2, 3, 4, 5)
1
(2, 3)
(3, 4, 5)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
(1, 2, 3, 4, 5, 6, 7, 8)
```

11. Declare a dictionary variable and perform the following task.

a. Print the value for 'one' key.

b. Print the value for 2 keys.

c. Print the complete dictionary.

d. Print all the keys.

e. Print all the values.

```python
o_dict = {'one': 1, 'two': 2, 'three': 3}

# a. Print the value for 'one' key.
print(o_dict['one'])

# b. Print the value for 2 keys.
print(o_dict['one'], o_dict['two'])

# c. Print the complete dictionary.
print(o_dict)

# d. Print all the keys.
print(o_dict.keys())

# e. Print all the values.
print(o_dict.values())

```

```
1
1 2
{'one': 1, 'two': 2, 'three': 3}
dict_keys(['one', 'two', 'three'])
dict_values([1, 2, 3])
```

12

```python
p_empty_set = set()
print(type(p_empty_set))

```

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

13

```python
list_sets = [{1, 2, 3}, {4, 5, 6}, {7, 8, 9}]
for s in list_sets:
    print(s)

```

```
{1, 2, 3}
{4, 5, 6}
{8, 9, 7}
```

14

```python
list_tuples = [(1, 2), (3, 4), (5, 6)]
for t in list_tuples:
    print(t)

```

```
(1, 2)
(3, 4)
(5, 6)
```

15

```python
list_dicts = [{'a': 1, 'b': 2}, {'c': 3, 'd': 4}, {'e': 5, 'f': 6}]
for d in list_dicts:
    print(d)

```

```
{'a': 1, 'b': 2}
{'c': 3, 'd': 4}
{'e': 5, 'f': 6}
```

16

```python
list_sets_tuples = [{1, 2}, (3, 4), {5, 6}, (7, 8)]
for i in list_sets_tuples:
    print(i)

```

```
{1, 2}
(3, 4)
{5, 6}
(7, 8)
```

17

```python
mixed_type_list = [{'a': 1, 'b': 2}, {1, 2, 3}, (1, 2, 3)]
for j in mixed_type_list:
    print(j)

```

```
{'a': 1, 'b': 2}
{1, 2, 3}
(1, 2, 3)
```

18

- List: Ordered collection of items that can be changed (mutable).
- Set: Collection of unique items with no duplicate elements and unordered (mutable).
- Tuple: Ordered collection of items that cannot be changed (immutable).
- Dictionary: Collection of key-value pairs, where keys are unique and immutable, and values can be mutable.

19

- We use a list datatype when we need an ordered collection of items that may change.
    - Set datatype is used when user needs to store unique items or perform set operations like union, intersection, etc.
    - Tuple is mainly used when the user needs an immutable ordered collection of items, especially for situations like returning multiple values from a function.
    - Dictionary is mostly used to store key-value pairs and look up values based on keys.

20

(i)

```python
# Convert integer to string
num_int = 1230
num_str = str(num_int)
print("Integer to String:", num_str, type(num_str))

# Convert string to integer
str_num = "4568"
str_int = int(str_num)
print("String to Integer:", str_int, type(str_int))

```

```
Integer to String: 1230 <class 'str'>
String to Integer: 4568 <class 'int'>
```

(ii)

```python
# Convert integer to float
num_int = 12374
num_float = float(num_int)
print("Integer to Float:", num_float, type(num_float))

```

```
Integer to Float: 12374.0 <class 'float'>
```

(iii)

```python
# Convert integer to string
num_int = 12388
num_str = str(num_int)
print("Integer to String:", num_str, type(num_str))

```

```
Integer to String: 12388 <class 'str'>
```

(iv)

```python
# Convert list to tuple
z_list = [1, 2, 3, 4, 5, 6, 7, 8]
z_tuple = tuple(z_list)
print("List to Tuple:", z_tuple, type(z_tuple))

```

```
List to Tuple: (1, 2, 3, 4, 5, 6, 7, 8) <class 'tuple'>
```

(v)

```python
# Convert tuple to list
u_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)
u_list = list(u_tuple)
print("Tuple to List:", u_list, type(u_list))

```

```
Tuple to List: [1, 2, 3, 4, 5, 6, 7, 8, 9] <class 'list'>
```

(vi)

```python
# Convert integer to character
ascii_value = 69
character = chr(ascii_value)
print("Integer to Character:", character, type(character))

```

```
Integer to Character: E <class 'str'>
```