In [1]:
```
tuple1 = (14, 52, 17, 24)
print(tuple1[1])
print(tuple1[3])
```

```
52
24
```

In [2]:
```
tuple1 = (14, 52, 17, 24)
print( len(tuple1) )
```

```
4
```

In [3]:
```
tuple1 = (14, 52, 17, 24)
for item in tuple1:
    print(item)
```

```
14
52
17
24
```

In [4]:
```
tuple1 = (14, 52, 17, 24)

index = 0
while index<len(tuple1):
    print(tuple1[index])
    index = index + 1
```

```
14
52
17
24
```

In [5]:
```
# Different types of tuples

# Empty tuple
my_tuple = ()
print(my_tuple)

# Tuple having integers
```

```
()
```

In [6]:
```
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)

# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)
```

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

In [7]:
```
my_tuple = 3, 4.6, "dog"
print(my_tuple)

# tuple unpacking is also possible
a, b, c = my_tuple

print(a)      # 3
print(b)      # 4.6
print(c)      # dog
```

```
(3, 4.6, 'dog')
3
4.6
dog
```

In [8]:
```
my_tuple = ("hello")
print(type(my_tuple))  # <class 'str'>

# Creating a tuple having one element
my_tuple = ("hello",)
print(type(my_tuple))  # <class 'tuple'>

# Parentheses is optional
my_tuple = "hello",
print(type(my_tuple))  # <class 'tuple'>
```

```
<class 'str'>
<class 'tuple'>
<class 'tuple'>
```

In [9]:
```
# Accessing tuple elements using indexing
my_tuple = ('p','e','r','m','i','t')

print(my_tuple[0])   # 'p'
print(my_tuple[5])   # 't'

# IndexError: list index out of range
# print(my_tuple[6])

# Index must be an integer
# TypeError: list indices must be integers, not float
# my_tuple[2.0]

# nested tuple
n_tuple = ("mouse", [8, 4, 6], (1, 2, 3))

# nested index
print(n_tuple[0][3])       # 's'
print(n_tuple[1][1])       # 4
```

```
p
t
s
4
```

In [10]:
```
# Accessing tuple elements using slicing
my_tuple = ('p','r','o','g','r','a','m','i','z')

# elements 2nd to 4th
# Output: ('r', 'o', 'g')
print(my_tuple[1:4])

# elements beginning to 2nd
# Output: ('p', 'r')
print(my_tuple[:-7])

# elements 8th to end
# Output: ('i', 'z')
print(my_tuple[7:])

# elements beginning to end
# Output: ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')
print(my_tuple[:])
```

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

In [11]:
```python
# Deleting tuples
my_tuple = ('p', 'r', 'o', 'g', 'r', 'a', 'm', 'i', 'z')

# can't delete items
# TypeError: 'tuple' object doesn't support item deletion
# del my_tuple[3]

# Can delete an entire tuple
del my_tuple

# NameError: name 'my_tuple' is not defined
print(my_tuple)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\SRINUP~1\AppData\Local\Temp/ipykernel_15168/252009680.py in <module>
     10 
     11 # NameError: name 'my_tuple' is not defined
---> 12 print(my_tuple)

NameError: name 'my_tuple' is not defined
```

In [12]:
```python
# Membership test in tuple
my_tuple = ('a', 'p', 'p', 'l', 'e',)

# In operation
print('a' in my_tuple)
print('b' in my_tuple)

# Not in operation
print('g' not in my_tuple)
```

```
True
False
True
```

In [13]:
```python
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}
```

In [15]:
```python
# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

In [ ]:

```
```

In [16]:

```
# get vs [] for retrieving elements
my_dict = {'name': 'Jack', 'age': 26}

# Output: Jack
print(my_dict['name'])

# Output: 26
print(my_dict.get('age'))

# Trying to access keys which doesn't exist throws error
# Output None
print(my_dict.get('address'))

# KeyError
print(my_dict['address'])
```

```
Jack
26
None

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\SRINUP~1\AppData\Local\Temp/ipykernel_15168/2429790587.py in <module>
     13 
     14 # KeyError
---> 15 print(my_dict['address'])

KeyError: 'address'
```

In [17]:

```
# Changing and adding Dictionary Elements
my_dict = {'name': 'Jack', 'age': 26}

# update value
my_dict['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}
print(my_dict)

# add item
my_dict['address'] = 'Downtown'
```

```
{'name': 'Jack', 'age': 27}
```

In [18]:

```
# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
print(my_dict)
```

```
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

In [19]:
```
# Removing elements from a dictionary

# create a dictionary
squares = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

# remove a particular item, returns its value
# Output: 16
print(squares.pop(4))

# Output: {1: 1, 2: 4, 3: 9, 5: 25}
print(squares)

# remove an arbitrary item, return (key,value)
# Output: (5, 25)
print(squares.popitem())

# Output: {1: 1, 2: 4, 3: 9}
print(squares)

# remove all items
squares.clear()

# Output: {}
print(squares)

# delete the dictionary itself
del squares

# Throws Error
print(squares)
```

```
16
{1: 1, 2: 4, 3: 9, 5: 25}
(5, 25)
{1: 1, 2: 4, 3: 9}
{}


---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:\Users\SRINUP~1\AppData\Local\Temp/ipykernel_15168/1968832105.py in <module>
     28
     29 # Throws Error
---> 30 print(squares)

NameError: name 'squares' is not defined
```

In [20]:
```
# Membership Test for Dictionary Keys
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: True
print(1 in squares)

# Output: True
print(2 not in squares)

# membership tests for key only not value
# Output: False
print(49 in squares)
```

```
True
True
False
```

In [21]:
```
# Dictionary Built-in Functions
squares = {0: 0, 1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

# Output: False
print(all(squares))

# Output: True
print(any(squares))

# Output: 6
print(len(squares))

# Output: [0, 1, 3, 5, 7, 9]
print(sorted(squares))
```

```
False
True
6
[0, 1, 3, 5, 7, 9]
```

In [22]:
```
# Different types of sets in Python
# set of integers
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
{1, 2, 3}
{1.0, 'Hello', (1, 2, 3)}
```

In [23]:
```
# set cannot have duplicates
# Output: {1, 2, 3, 4}
my_set = {1, 2, 3, 4, 3, 2}
print(my_set)

# we can make set from a list
# Output: {1, 2, 3}
my_set = set([1, 2, 3, 2])
print(my_set)

# set cannot have mutable items
# here [3, 4] is a mutable list
# this will cause an error.

my_set = {1, 2, [3, 4]}
```

```
{1, 2, 3, 4}
{1, 2, 3}

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\SRINUP~1\AppData\Local\Temp/ipykernel_15168/919206453.py in <module>
     13 # this will cause an error.
     14 
---> 15 my_set = {1, 2, [3, 4]}

TypeError: unhashable type: 'list'
```

In [24]:
```
# Distinguish set and dictionary while creating empty set

# initialize a with {}
a = {}

# check data type of a
print(type(a))

# initialize a with set()
a = set()

# check data type of a
print(type(a))
```

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

In [25]:
```
# initialize my_set
my_set = {1, 3}
print(my_set)

# my_set[0]
# if you uncomment the above line
# you will get an error
# TypeError: 'set' object does not support indexing

# add an element
# Output: {1, 2, 3}
my_set.add(2)
print(my_set)

# add multiple elements
# Output: {1, 2, 3, 4}
my_set.update([2, 3, 4])
print(my_set)

# add list and set
# Output: {1, 2, 3, 4, 5, 6, 8}
my_set.update([4, 5], {1, 6, 8})
print(my_set)
```

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

In [26]:
```
#lambda function
square = lambda a: a*a
#call lambda function
result = square(6)
print(result)
```

```
36
```

In [27]:
```
#lambda function
mul = lambda a,b: a*b
#call lambda function
result = mul(5,3)
print(result)
```

```
15
```

In [28]:
```
#lambda function
six = lambda : 6
#call lambda function
result = six()
print(result)
```

```
6
```

In [29]:
```
#recursive lambda function
factorial = lambda a: a*factorial(a-1) if (a>1) else 1
#call lambda function
result = factorial(5)
print(result)
```

```
120
```

In [30]:
```
import math
#function returning lambda function
def myfunc(n):
  return lambda a : math.pow(a, n)

#lambda functions
square = myfunc(2) #square = lambda a : math.pow(a, 2)
cube = myfunc(3) #cube =  = lambda a : math.pow(a, 3)
squareroot = myfunc(0.5) #squareroot = lambda a : math.pow(a, 0.5)

print(square(3))
print(cube(3))
print(squareroot(3))
```

```
9.0
27.0
1.7320508075688772
```

In [31]:
```
Program to double each item in a list using map()

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(map(lambda x: x * 2 , my_list))

print(new_list)
```

```
  File "C:\Users\SRINUP~1\AppData\Local\Temp/ipykernel_15168/2051395365.py", li
ne 1
    Program to double each item in a list using map()
            ^
SyntaxError: invalid syntax
```

In [ ]: