Tuple

In [1]:
```
t=()
t
```

Out[1]:
```
()
```

In [2]:
```
type(t)
```

Out[2]:
```
tuple
```

In [8]:
```
t=(10,20,30)
t
```

Out[8]:
```
(10, 20, 30)
```

In [9]:
```
t.count(20)
```

Out[9]:
```
1
```

In [10]:
```
t1=(10,20,2.2,'ten',True,1+2j)
t1
```

Out[10]:
```
(10, 20, 2.2, 'ten', True, (1+2j))
```

In [11]:
```
t1.count(20)
```

Out[11]:
```
1
```

In [12]:
```
t1.index(20)
```

Out[12]:
```
1
```

In [13]:
```
print(t)
```

```
print (t1)
```

```
(10, 20, 30)
(10, 20, 2.2, 'ten', True, (1+2j))
```

In [14]:
```
print(len(t))
print(len(t1))
```

```
3
6
```

In [15]:
```
t
```

Out[15]: (10, 20, 30)

In [16]:
```
t[0]
```

Out[16]: 10

In [17]:
```
t[0]
```

Out[17]: 10

In [18]:
```
t=[100]
```

In [19]:
```
t
```

Out[19]: [100]

In [26]:
```
t[0]=100
```

In [27]:
```
t
```

Out[27]: [100]

In [28]:
```
bank_account=(1234,'sbin00',10000)
bank_account
```

Out[28]: (1234, 'sbin00', 10000)

In [29]:
```
bank_account[2]=20000
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 bank_account[2]=20000

TypeError: 'tuple' object does not support item assignment
```

In [30]:
```
t
```

Out[30]:
```
[100]
```

In [31]:
```
# the number are repeat not any change
t2=t*3
t2
```

Out[31]:
```
[100, 100, 100]
```

In [32]:
```
t
```

Out[32]:
```
[100]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
# list
-mutable
-duplicate is allowed
-append(),copy(),insert(),extend(),pop()
-remove the element
-list is growble
-multiple data type in a list
```

```
-indexing & slicing is allowed


# Tuple
-immutable(unchangeable)
- duplication is allowed
- remove the
-
```

set

In [33]:
```
s={}
s
```

Out[33]: {}

In [34]:
```
type(s)
```

Out[34]: dict

In [35]:
```
s1=set()
s1
```

Out[35]: set()

In [36]:
```
s2={90,10,50,80,40,25}
s2
```

Out[36]: {10, 25, 40, 50, 80, 90}

In [37]:
```
s2={90,10,50,80,40,25,10}
s2
```

Out[37]: {10, 25, 40, 50, 80, 90}

In [38]:
```
type(s2)
```

Out[38]: set

In [39]:
```
s2
```

Out[39]: {10, 25, 40, 50, 80, 90}

In [40]:
```
s3=s2.copy()
s3
```

Out[40]: {10, 25, 40, 50, 80, 90}

In [41]:
```
s3
```

Out[41]: {10, 25, 40, 50, 80, 90}

In [42]:
```
s3.add(3.4)
```

In [43]:
```
s3
```

Out[43]: {3.4, 10, 25, 40, 50, 80, 90}

In [45]:
```
s3.add('nit')
```

In [46]:
```
s3
```

Out[46]: {10, 25, 3.4, 40, 50, 80, 90, 'nit'}

In [47]:
```
s3.add(1+2j)
s3.add(True)
```

In [48]:
```
s3
```

Out[48]: {(1+2j), 10, 25, 3.4, 40, 50, 80, 90, True, 'nit'}

In [49]:
```
print(s)
print(s1)
```

```
print(s2)
print(s3)
```

```
{}
set()
{80, 50, 90, 40, 25, 10}
{'nit', True, 3.4, (1+2j), 10, 80, 25, 90, 40, 50}
```

In [50]:
```
s
```

Out[50]:
```
{}
```

In [51]:
```
type(s)
```

Out[51]:
```
dict
```

In [52]:
```
s3
```

Out[52]:
```
{(1+2j), 10, 25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [53]:
```
s3.remove(2000)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[53], line 1
----> 1 s3.remove(2000)

KeyError: 2000
```

In [54]:
```
s3.remove(1+2j)
```

In [55]:
```
s3
```

Out[55]:
```
{10, 25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [56]:
```
s3
```

Out[56]:
```
{10, 25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [59]:
```
s3.discard(2000)
```

In [60]:
```
s3.discard(10)
```

In [61]:
```
s3
```

Out[61]:
```
{25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [58]:
```
s3
```

Out[58]:
```
{10, 25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [62]:
```
s3
```

Out[62]:
```
{25, 3.4, 40, 50, 80, 90, True, 'nit'}
```

In [63]:
```
s3.pop()
```

Out[63]:
```
'nit'
```

In [64]:
```
s3
```

Out[64]:
```
{True, 3.4, 25, 40, 50, 80, 90}
```

In [65]:
```
s3.pop()
```

Out[65]:
```
True
```

In [66]:
```
s3
```

Out[66]:
```
{3.4, 25, 40, 50, 80, 90}
```

In [67]:
```
s3.pop(0)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[67], line 1
----> 1 s3.pop(0)

TypeError: set.pop() takes no arguments (1 given)
```

In [68]:
```
s3
```

Out[68]:
```
{3.4, 25, 40, 50, 80, 90}
```

In [69]:
```
s3[:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[69], line 1
----> 1 s3[:]

TypeError: 'set' object is not subscriptable
```

In [70]:
```
s3[1:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[70], line 1
----> 1 s3[1:]

TypeError: 'set' object is not subscriptable
```

In [71]:
```
s3
```

Out[71]:
```
{3.4, 25, 40, 50, 80, 90}
```

In [72]:
```
s3[2]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[72], line 1
----> 1 s3[2]

TypeError: 'set' object is not subscriptable
```

In [73]:
```
s3
```

Out[73]:
```
{3.4, 25, 40, 50, 80, 90}
```

In [74]:
```
s3.pop(0)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[74], line 1
----> 1 s3.pop(0)

TypeError: set.pop() takes no arguments (1 given)
```

In [75]:
```
s3.pop()
```

Out[75]:
```
3.4
```

In [76]:
```
40 in s3
```

Out[76]:
```
True
```

In [79]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
c={8,9,10}
```

In [80]:
```
type(a)
```

Out[80]:
```
set
```

In [81]:
```
a.union(b)
```

Out[81]: {1, 2, 3, 4, 5, 6, 7, 8}

In [82]:
```
a.union(b,c)
```

Out[82]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [84]:
```
print(a)
print(b)
print(c)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [83]:
```
a|b
```

Out[83]: {1, 2, 3, 4, 5, 6, 7, 8}

In [88]:
```
b|c
```

Out[88]: {4, 5, 6, 7, 8, 9, 10}

In [85]:
```
a|b|c
```

Out[85]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [86]:
```
a|c
```

Out[86]: {1, 2, 3, 4, 5, 8, 9, 10}

In [87]:
```
a|c|b
```

Out[87]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

intersection

In [89]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
c={8,9,10}
```

In [90]:
```
a.intersection(b)
```

Out[90]: {4, 5}

In [91]:
```
b.intersection(c)
```

Out[91]: {8}

In [92]:
```
a.intersection(c)
```

Out[92]: set()

In [93]:
```
a & b
```

Out[93]: {4, 5}

In [94]:
```
b&c
```

Out[94]: {8}

Difference

In [9]:
```
a={1,2,3,4,5}
b={4,5,6,7,8}
c={8,9,10}
```

In [10]:
```
a.difference(b)
```

Out[10]: {1, 2, 3}

In [11]:
```
b.difference(a)
```

Out[11]: {6, 7, 8}

In [12]:
```
c.difference(b)
```

Out[12]: {9, 10}

In [13]:
```
b-c
```

Out[13]: {4, 5, 6, 7}

In [14]:
```
c-b
```

Out[14]: {9, 10}

In [15]:
```
a-b-c
```

Out[15]: {1, 2, 3}

In [16]:
```
print(a)
print(b)
print(c)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [17]:
```
a.symmetric_difference(b)
```

Out[17]: {1, 2, 3, 6, 7, 8}

In [18]:
```
b.symmetric_difference(c)
```

Out[18]: {4, 5, 6, 7, 9, 10}

In [19]:
```
a.symmetric_difference(c)
```

Out[19]: {1, 2, 3, 4, 5, 8, 9, 10}

In [20]:
```
print(a)
print(b)
print(c)
```

```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [21]:
```
a.symmetric_difference_update(b)
```

In [22]:
```
a
```

Out[22]:
```
{1, 2, 3, 6, 7, 8}
```

In [23]:
```
print(a)
print(b)
print(c)
```

```
{1, 2, 3, 6, 7, 8}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

Superset,Subset,Disjoint operation

In [31]:
```
s4={1,2,3,4,5,6,7,8,9}
s5={3,4,5,6,7,8,}
s6={10,20,30,40}
```

In [32]:
```
s4.issuperset(s5)
```

Out[32]:
```
True
```

In [27]:
```
s5.issubset(s4)
```

Out[27]:
```
True
```

In [28]:
```
s6.isdisjoint(s4)
```

Out[28]: True

In [29]:
```
s6.issubset(s5)
```

Out[29]: False

In [30]:
```
s6.issubset(s4)
```

Out[30]: False

In [37]:
```
s7={1,2,3,4,5,6,7,8,9}
s8={15,25,35}
s9={10,20,30,40}
```

In [38]:
```
s7.issuperset(s8)
```

Out[38]: False

In [39]:
```
s8.issubset(s7)
```

Out[39]: False

In [40]:
```
s7.isdisjoint(s8)
```

Out[40]: True

Python Dictionary

In [41]:
```
d={}
d
```

Out[41]: {}

In [42]:
```
type(d)
```

Out[42]: dict

In [69]:
```
d1= {1: 'one',2: 'two',3: 'three', 'four':4,'l':[1,2,3]}
```

In [70]:
```
d1
```

Out[70]: {1: 'one', 2: 'two', 3: 'three', 'four': 4, 'l': [1, 2, 3]}

In [71]:
```
d2=d1.copy()
```

In [72]:
```
d2
```

Out[72]: {1: 'one', 2: 'two', 3: 'three', 'four': 4, 'l': [1, 2, 3]}

In [73]:
```
d1.items()
```

Out[73]: dict_items([(1, 'one'), (2, 'two'), (3, 'three'), ('four', 4), ('l', [1, 2, 3])])

In [74]:
```
len(d1.items())
```

Out[74]: 5

In [75]:
```
d1
```

Out[75]: {1: 'one', 2: 'two', 3: 'three', 'four': 4, 'l': [1, 2, 3]}

In [76]:
```
d1[1]
```

Out[76]: 'one'

In [77]:
```
d1[2]
```

Out[77]: 'two'

In [78]:
```
d1[3]
```

Out[78]: 'three'

In [79]:
```
d1['four']
```

Out[79]:
```
4
```

In [80]:
```
d1[1]
```

Out[80]:
```
'one'
```

In [81]:
```
d1.keys()
```

Out[81]:
```
dict_keys([1, 2, 3, 'four', 'l'])
```

In [82]:
```
d1.values()
```

Out[82]:
```
dict_values(['one', 'two', 'three', 4, [1, 2, 3]])
```

In [83]:
```
d1
```

Out[83]:
```
{1: 'one', 2: 'two', 3: 'three', 'four': 4, 'l': [1, 2, 3]}
```

In [86]:
```
d1
```

Out[86]:
```
{1: 'one', 2: 'two', 3: 'three', 'four': 4, 'l': [1, 2, 3]}
```

In [87]:
```
d1.pop('l')
```

Out[87]:
```
[1, 2, 3]
```

In [88]:
```
1 in d1
```

Out[88]:
```
True
```

In [90]:
```
100 in d1
```

Out[90]:
```
False
```

Range

In [1]:
```
range(20)
```

Out[1]: range(0, 20)

In [2]:
```
range(20,30)
```

Out[2]: range(20, 30)

In [3]:
```
range(20,30,5)
```

Out[3]: range(20, 30, 5)

In [4]:
```
list(range(20))
```

Out[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

In [5]:
```
list(range(20,30))
```

Out[5]: [20, 21, 22, 23, 24, 25, 26, 27, 28, 29]

In [6]:
```
list(range(20,30,5))
```

Out[6]: [20, 25]

In [7]:
```
r=range(20,30,5)
r
```

Out[7]: range(20, 30, 5)

In [8]:
```
for i in r:
    print(i)
```

```
20
25
```

Help Function

In [1]:
```
help()
```

```
Welcome to Python 3.13's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.13/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q", "quit" or "exit".
```

```
Here is a list of available topics.  Enter any topic name to get more help.

ASSERTION                EXCEPTIONS               PACKAGES
ASSIGNMENT               EXECUTION                POWER
ASSIGNMENTEXPRESSIONS    EXPRESSIONS              PRECEDENCE
ATTRIBUTEMETHODS         FLOAT                    PRIVATENAMES
ATTRIBUTES               FORMATTING               RETURNING
AUGMENTEDASSIGNMENT      FRAMEOBJECTS             SCOPING
BASICMETHODS             FRAMES                   SEQUENCEMETHODS
BINARY                   FUNCTIONS                SEQUENCES
BITWISE                  IDENTIFIERS              SHIFTING
BOOLEAN                  IMPORTING                SLICINGS
CALLABLEMETHODS          INTEGER                  SPECIALATTRIBUTES
CALLS                    LISTLITERALS             SPECIALIDENTIFIERS
CLASSES                  LISTS                    SPECIALMETHODS
CODEOBJECTS              LITERALS                 STRINGMETHODS
COMPARISON               LOOPING                  STRINGS
COMPLEX                  MAPPINGMETHODS           SUBSCRIPTS
CONDITIONAL              MAPPINGS                 TRACEBACKS
CONTEXTMANAGERS          METHODS                  TRUTHVALUE
CONVERSIONS              MODULES                  TUPLELITERALS
DEBUGGING                NAMESPACES               TUPLES
DELETION                 NONE                     TYPEOBJECTS
DICTIONARIES             NUMBERMETHODS            TYPES
DICTIONARYLITERALS       NUMBERS                  UNARY
DYNAMICFEATURES          OBJECTS                  UNICODE
ELLIPSIS                 OPERATORS

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

```
You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
```

In [2]:
```
pi=3.14
pi
```

Out[2]: 3.14

In [3]:
```
pi=3.15
pi
```

Out[3]: 3.15

In [4]:
```
a=5
b=6
```

In [6]:
```
a=b
b=a
```

In [7]:
```
print(a)
print(b)
```

```
6
6
```

In [8]:
```
a1=7
b1=8
```

In [9]:
```
temp=a1
a1=b1
b1=temp
```

In [10]:
```
print(a1)
print(b1)
```

```
8
7
```

In [11]:
```
a2=5
b2=6
```

In [12]:
```
a2=a2+b2
b2=a2-b2
a2=a2-b2
```

In [13]:
```
print(a2)
print(b2)
```

```
6
5
```

import math function

In [ ]:
```
. add()
. sub()
. mul()
```

In [ ]:
```
Rename a string "fine" to "dine"
```

In [2]:
```
a='fine'
b=a[0]
b='d'
```

In [3]:
```
b
```

Out[3]:
```
'd'
```

In [4]:
```
b+'ine'
```

Out[4]:
```
'dine'
```

In [5]:
```
import math as m
m.sqrt(25)
```

Out[5]: 5.0

In [ ]:
```
floor()---gives the minimum value
```

In [8]:
```
m.floor(2.9)  # Gives the minimum value
```

Out[8]: 2

In [ ]:
```
ceil()---- Gives the maximum value
```

In [11]:
```
m.ceil(2.5)  # gives the maximum value
```

Out[11]: 3

In [ ]:
```
pow()--- Gives the Exponantion of value
```

In [12]:
```
m.pow(3,2)
```

Out[12]: 9.0

In [14]:
```
m.pi
```

Out[14]: 3.141592653589793

In [18]:
```
m.e
```

Out[18]: 2.718281828459045

In [16]:
```
round(pow(3))
```

Out[16]: 729

In [20]:
```
round(4.6)
```

Out[20]: 5

In [21]:
```
round(4,6)
```

Out[21]: 4

In [ ]: