Jumping statements in Python

In [1]:

```
# break
# continue
# pass
```

In [3]:

```
name = "Yogesh Bhimrao Rakate"
for i in name:
    print(i)
    if i == 'm':
        break
```

```
Y
o
g
e
s
h

B
h
i
m
```

In [5]:

```
for i in range(1,20):
    print(i)
    if i == 7:
        break
```

```
1
2
3
4
5
6
7
```

Break terminates the loop at particular point

continue skips the statement in the loop

In [7]:

```
for i in name:
    if i == 'm' or i == 'h':
        continue
    print(i)
```

```
Y
o
g
e
s

B
i
r
a
o

R
a
k
a
t
e
```

In [11]:

```
for j in range(1,30):
    if j in range(10, 15):
        continue
    print(j, end=' ')
```

```
1 2 3 4 5 6 7 8 9 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

In [13]:

```
#prime number
num=5
flag = False
for k in range(2,num):
    if num%k == 0:
        flag = True
        break
if flag:
    print("Not a prime number")
else:
    print("Prime number")
```

```
Prime number
```

In [16]:

```
# pass
for j in range(1,5):
    pass
```

In [20]:

```
if flag:
    pass
else:
    pass
```

Predefined data structures in Python

In [21]:

```
a = []
b = ()
c = {}
d = {2}
```

In [22]:

```
type(a)
```

Out[22]:

```
list
```

List is a collection of heterogeneous elements

In [23]:

```
type(b)
```

Out[23]:

```
tuple
```

Tuple is also a kind of heterogeneous collection (not modifiable) immutable

In [24]:

```
type(c)
```

Out[24]:

```
dict
```

Dictionary is the collection of key and value pair

In [25]:

```
type(d)
```

Out[25]:

```
set
```

Set is a collection of unique elements

List and Tuple

In [26]:

```
a = [1,2,3,4,5]
b = (1,2,3,4,5)
```

In [27]:

```
a
```

Out[27]:

```
[1, 2, 3, 4, 5]
```

In [28]:

```
b
```

Out[28]:

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

In [29]:

```
len(a)
```

Out[29]:

```
5
```

In [30]:

```
len(b)
```

Out[30]:

```
5
```

In [31]:

```
sum(a)
```

Out[31]:

```
15
```

In [32]:

```
sum(b)
```

Out[32]:

```
15
```

In [33]:

```
max(a)
```

Out[33]:

```
5
```

In [34]:

```
min(b)
```

Out[34]:

```
1
```

In [35]:

```
a[0]
```

Out[35]:

```
1
```

In [36]:

```
b[0]
```

Out[36]:

```
1
```

In [37]:

```
a[1:3]
```

Out[37]:

```
[2, 3]
```

In [38]:

```
b[1:3]
```

Out[38]:

```
(2, 3)
```

In [39]:

```
a[2] = 'Yogesh'
```

In [40]:

```
a
```

Out[40]:

```
[1, 2, 'Yogesh', 4, 5]
```

In [41]:

```
b[2] = 'Yogesh'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [41], line 1
----> 1 b[2] = 'Yogesh'

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

In [42]:

```
# Examples for tuple are Location co-ordinates and Employee id
```

In [43]:

```
b
```

Out[43]:

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

In [44]:

```
b.index(4)
```

Out[44]:

```
3
```

In [45]:

```
b.count(2)
```

Out[45]:

```
1
```

In [46]:

```
a.append("Rakate")
```

In [47]:

```
a
```

Out[47]:

```
[1, 2, 'Yogesh', 4, 5, 'Rakate']
```

In [48]:

```
a.insert(4,'Bhimrao')
```

In [49]:

```
a
```

Out[49]:

```
[1, 2, 'Yogesh', 4, 'Bhimrao', 5, 'Rakate']
```

In [50]:

```
a.pop()
```

Out[50]:

```
'Rakate'
```

In [51]:

```
a
```

Out[51]:

```
[1, 2, 'Yogesh', 4, 'Bhimrao', 5]
```

In [52]:

```
a.pop(4)
```

Out[52]:

```
'Bhimrao'
```

In [53]:

```
a
```

Out[53]:

```
[1, 2, 'Yogesh', 4, 5]
```

In [54]:

```
a.remove('Yogesh')
```

In [55]:

```
a
```

Out[55]:

```
[1, 2, 4, 5]
```

In [56]:

```
e = [7,3,6,0]
```

In [57]:

```
a+e
```

Out[57]:

```
[1, 2, 4, 5, 7, 3, 6, 0]
```

In [58]:

```
a
```

Out[58]:

```
[1, 2, 4, 5]
```

In [59]:

```
a.extend(e)
```

In [60]:

```
a
```

Out[60]:

```
[1, 2, 4, 5, 7, 3, 6, 0]
```

In [61]:

```
f = (10,8,9)
```

In [62]:

```
a.extend(f)
```

In [63]:

```
a
```

Out[63]:

```
[1, 2, 4, 5, 7, 3, 6, 0, 10, 8, 9]
```

In [64]:

```
a.sort()
```

In [65]:

```
a
```

Out[65]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [66]:

```
a.sort(reverse=True)
```

In [67]:

```
a
```

Out[67]:

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

In [68]:

```
#Remind the key parameter after functions topic
```

In [69]:

```
a.reverse()
```

In [70]:

```
a
```

Out[70]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [71]:

```
yogesh = a
soniya = a.copy()
```

In [72]:

```
id(yogesh)
```

Out[72]:

```
1419984471296
```

In [73]:

```
id(a)
```

Out[73]:

```
1419984471296
```

In [74]:

```
id(soniya)
```

Out[74]:

```
1420012620608
```

a.copy is a shallow copy

In [75]:

```
a.clear()
```

In [76]:

```
a
```

Out[76]:

```
[]
```

In [77]:

```
yogesh
```

Out[77]:

```
[]
```

In [78]:

```
soniya
```

Out[78]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [79]:

```
tuple(soniya)
```

Out[79]:

```
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

In [80]:

```
set(soniya)
```

Out[80]:

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [81]:

```
help(list)
```

```
|  The argument must be an iterable if specified.
|
|  Methods defined here:
|
|  __add__(self, value, /)
|      Return self+value.
|
|  __contains__(self, key, /)
|      Return key in self.
|
|  __delitem__(self, key, /)
|      Delete self[key].
|
|  __eq__(self, value, /)
|      Return self==value.
|
|  __ge__(self, value, /)
|      Return self>=value.
|
|  __getattribute__(self, name, /)
```

Dictionary

In [83]:

```
soniya[5]
```

Out[83]:

```
5
```

Dictionary is accessible on the basis of key whereas list is accessible on the basis of index

In [85]:

```
for i in soniya:
    if i%2==0:
        print(i)
```

```
0
2
4
6
8
10
```

In [86]:

```
yogesh = {}
```

In [88]:

```
yogesh[0] = 'Bhimrao'
```

In [89]:

```
yogesh['dept'] = 'digital transformation'
```

In [90]:

```
yogesh
```

Out[90]:

```
{0: 'Bhimrao', 'dept': 'digital transformation'}
```

In [91]:

```
yogesh[0]
```

Out[91]:

```
'Bhimrao'
```

In [93]:

```
yogesh['dept']
```

Out[93]:

```
'digital transformation'
```

In [94]:

```
yogesh['dept'] = 'python'
```

In [95]:

```
yogesh
```

Out[95]:

```
{0: 'Bhimrao', 'dept': 'python'}
```

In [96]:

```
yogesh['salary'] = 100000
```

In [97]:

```
yogesh
```

Out[97]:

```
{0: 'Bhimrao', 'dept': 'python', 'salary': 100000}
```

In [98]:

```
employee = {'name':'Yogesh', 'gender':'Male', 'city':'Sangli'}
```

In [99]:

```
employee
```

Out[99]:

```
{'name': 'Yogesh', 'gender': 'Male', 'city': 'Sangli'}
```

In [101]:

```
for i in yogesh:
    print(i,yogesh[i])
```

```
0 Bhimrao
dept python
salary 100000
```

In [103]:

```
for j in employee:
    print(j, employee[j])
```

```
name Yogesh
gender Male
city Sangli
```

In [104]:

```
fruit = [['apple', 40], ['Banana', 5], ['mango', 50], ['guava', 20]]
```

In [129]:

```
dfruit = {}
for i in fruit:
    dfruit[i[0]] = i[1]
```

In [130]:

```
dfruit
```

Out[130]:

```
{'apple': 40, 'Banana': 5, 'mango': 50, 'guava': 20}
```

In [131]:

```
dict(fruit)
```

Out[131]:

```
{'apple': 40, 'Banana': 5, 'mango': 50, 'guava': 20}
```

In [132]:

```
dict(soniya)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [132], line 1
----> 1 dict(soniya)

TypeError: cannot convert dictionary update sequence element #0 to a sequenc
e
```

In [133]:

```
for i,j in fruit:
    print(i,j)
```

```
apple 40
Banana 5
mango 50
guava 20
```

In [134]:

```
yogesh.keys()
```

Out[134]:

```
dict_keys([0, 'dept', 'salary'])
```

In [135]:

```
employee.keys()
```

Out[135]:

```
dict_keys(['name', 'gender', 'city'])
```

In [136]:

```
dfruit.keys()
```

Out[136]:

```
dict_keys(['apple', 'Banana', 'mango', 'guava'])
```

In [137]:

```
dfruit.fromkeys(soniya)
```

Out[137]:

```
{0: None,
 1: None,
 2: None,
 3: None,
 4: None,
 5: None,
 6: None,
 7: None,
 8: None,
 9: None,
 10: None}
```

In [138]:

```
dfruit.fromkeys(soniya, 'Hello')
```

Out[138]:

```
{0: 'Hello',
 1: 'Hello',
 2: 'Hello',
 3: 'Hello',
 4: 'Hello',
 5: 'Hello',
 6: 'Hello',
 7: 'Hello',
 8: 'Hello',
 9: 'Hello',
 10: 'Hello'}
```

In [139]:

```
dfruit.get('apple')
```

Out[139]:

```
40
```

In [140]:

```
dfruit.items()
```

Out[140]:

```
dict_items([('apple', 40), ('Banana', 5), ('mango', 50), ('guava', 20)])
```

In [141]:

```
dfruit.pop('guava')
```

Out[141]:

```
20
```

In [142]:

```
dfruit
```

Out[142]:

```
{'apple': 40, 'Banana': 5, 'mango': 50}
```

In [143]:

```
dfruit.popitem()
```

Out[143]:

```
('mango', 50)
```

In [151]:

```
dfruit.setdefault('company','50')
```

Out[151]:

```
'50'
```

In [152]:

```
dfruit
```

Out[152]:

```
{'apple': 40, 'Banana': 5, 'starfruit': None, 'company': '50'}
```

In [155]:

```
dfruit.setdefault('program', 'java')
```

Out[155]:

```
'python'
```

In [156]:

```
dfruit
```

Out[156]:

```
{'apple': 40,
 'Banana': 5,
 'starfruit': None,
 'company': '50',
 'program': 'python'}
```

In [157]:

```
dfruit.update({'starfruit':50, 'company':'MouriTech'})
```

In [158]:

```
dfruit
```

Out[158]:

```
{'apple': 40,
 'Banana': 5,
 'starfruit': 50,
 'company': 'MouriTech',
 'program': 'python'}
```

In [159]:

```
dfruit['starfruit'],dfruit['company'] = 60,'MouriTech Ltd'
```

In [160]:

```
dfruit
```

Out[160]:

```
{'apple': 40,
 'Banana': 5,
 'starfruit': 60,
 'company': 'MouriTech Ltd',
 'program': 'python'}
```

In [161]:

```
dfruit.values()
```

Out[161]:

```
dict_values([40, 5, 60, 'MouriTech Ltd', 'python'])
```

In [162]:

```
dfruit.clear()
```

In [163]:

```
dfruit
```

Out[163]:

```
{}
```

In [164]:

```
len(dfruit)
```

Out[164]:

```
0
```

In [167]:

```
for i in employee:
    print(i, employee[i])
```

```
name Yogesh
gender Male
city Sangli
```

In [165]:

```
help(dict)
```

```
 |      See PEP 585
 |
 |  fromkeys(iterable, value=None, /) from builtins.type
 |      Create a new dictionary with keys from iterable and values set to
value.
 |
 |  ------------------------------------------------------------------
-
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate sign
ature.
 |
 |  ------------------------------------------------------------------
-
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

In [172]:

```
for i,j in employee.items():
    print(i,j)
```

```
name Yogesh
gender Male
city Sangli
```

Set

In [169]:

```
d={1,2,3,4,5,6,7,8,2,4,52,3,5}
```

In [170]:

```
d
```

Out[170]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 52}
```

In [173]:

```
e={5,6,7,8,9,10}
```

In [174]:

```
d.add(44)
```

In [175]:

```
d
```

Out[175]:

```
{1, 2, 3, 4, 5, 6, 7, 8, 44, 52}
```

In [176]:

```
d.difference(e)
```

Out[176]:

```
{1, 2, 3, 4, 44, 52}
```

In [177]:

```
e.difference(d)
```

Out[177]:

```
{9, 10}
```

In [178]:

```
d.intersection(e)
```

Out[178]:

```
{5, 6, 7, 8}
```

In [180]:

```
d.isdisjoint(e)
```

Out[180]:

```
False
```

In [181]:

```
f={3, 4, 5}
g={4}
f.isdisjoint(g)
```

Out[181]:

```
True
```

In [182]:

```
d.issubset(e)
```

Out[182]:

```
False
```

In [185]:

```
f.issubset(d)
```

Out[185]:

```
True
```

In [186]:

```
d.issuperset(f)
```

Out[186]:

```
True
```

In [187]:

```
f.issuperset(d)
```

Out[187]:

```
False
```

In [188]:

```
d.pop()
```

Out[188]:

```
1
```

In [189]:

```
d
```

Out[189]:

```
{2, 3, 4, 5, 6, 7, 8, 44, 52}
```

In [190]:

```
d.remove(44)
```

In [191]:

```
d
```

Out[191]:

```
{2, 3, 4, 5, 6, 7, 8, 52}
```

In [192]:

```
d.symmetric_difference(e)
```

Out[192]:

```
{2, 3, 4, 9, 10, 52}
```

In [193]:

```
d, e
```

Out[193]:

```
({2, 3, 4, 5, 6, 7, 8, 52}, {5, 6, 7, 8, 9, 10})
```

In [194]:

```
d.union(e)
```

Out[194]:

```
{2, 3, 4, 5, 6, 7, 8, 9, 10, 52}
```

In [195]:

```
d.difference(e)
```

Out[195]:

```
{2, 3, 4, 52}
```

In [196]:

```
d
```

Out[196]:

```
{2, 3, 4, 5, 6, 7, 8, 52}
```

In [197]:

```
d.difference_update(e)
```

In [198]:

```
d
```

Out[198]:

```
{2, 3, 4, 52}
```

In [199]:

```
d.intersection_update(e)
```

In [200]:

```
d
```

Out[200]:

```
set()
```

In [201]:

```
e
```

Out[201]:

```
{5, 6, 7, 8, 9, 10}
```

In [202]:

```
d.update(e)
```

In [203]:

```
d
```

Out[203]:

```
{5, 6, 7, 8, 9, 10}
```

In [209]:

```
name = ('Soniya', 'Abhishek', 'Hrushikesh', 'Nitin')
```

In [216]:

```
','.join(name)
```

Out[216]:

```
'Soniya,Abhishek,Hrushikesh,Nitin'
```

In [211]:

```
list('Yogesh')
```

Out[211]:

```
['Y', 'o', 'g', 'e', 's', 'h']
```

In [214]:

```
'Yogesh likes coding'.split(' ')
```

Out[214]:

```
['Yogesh', 'likes', 'coding']
```

In [218]:

```
num = [1,2,3,4]
snum = []
for i in num:
    snum.append(str(i))
```

In [219]:

```
snum
```

Out[219]:

```
['1', '2', '3', '4']
```

In [220]:

```
''.join(snum)
```

Out[220]:

```
'1234'
```

In [221]:

```
# remove the duplicate elements from the list
lis = [2,4,5,3,2,5,7,4]
```

In [223]:

```
list(set(lis))
```

Out[223]:

```
[2, 3, 4, 5, 7]
```

In [224]:

```
# another way
d=[]
for i in lis:
    if i not in d:
        d.append(i)
```

In [225]:

```
d
```

Out[225]:

```
[2, 4, 5, 3, 7]
```

In [226]:

```
e = []
for i in lis:
    if e.count(i) == 0:
        e.append(i)
```

In [227]:

```
e
```

Out[227]:

```
[2, 4, 5, 3, 7]
```

In [228]:

```
# another way
import collections
```

In [230]:

```
list(collections.OrderedDict.fromkeys(lis))
```

Out[230]:

```
[2, 4, 5, 3, 7]
```

In [232]:

```
list(enumerate(lis))
```

Out[232]:

```
[(0, 2), (1, 4), (2, 5), (3, 3), (4, 2), (5, 5), (6, 7), (7, 4)]
```

In [233]:

```
for i in enumerate(lis):
    print(i)
```

```
(0, 2)
(1, 4)
(2, 5)
(3, 3)
(4, 2)
(5, 5)
(6, 7)
(7, 4)
```

In [234]:

```
#find pilindrome strings in a given sentence
```

In [239]:

```
z='Malayalam is difficult language Soniya madam said this.'
cz = z.split()
c=0
for i in cz:
    if i.lower() == i.lower()[::-1]:
        c+=1
print(c)
```

```
2
```

In [242]:

```
#Validate the ip address
ip = input().split(':')
if 0 <= int(ip[0]) <=255 and 0 <= int(ip[1]) <=255 and 0 <= int(ip[2]) <=255 and 0 <= int(i
    print("valid ip address")
else:
    print('invalid ip address')
```

```
1234:253:2:3
invalid ip address
```

In [ ]:

```

```