```
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 10 16:44:35 2015
@author: hina
Reference: https://docs.python.org/3/tutorial/index.html
"""

print ()

# Tuples ()
# Tuples are immutable sequences, typically used to store collections of heterogeneous
# data, or for cases where an immutable sequence of homogeneous data is needed

# declaring empty tuples

tuple1 = ()
print ("tuple1: ", tuple1, len(tuple1))

tuple2 = tuple ()
print ("tuple2: ", tuple2, len(tuple2))

print ()

# singleton tuples must have a trailing comma

tuple3 = (1,) # tuple3 = (1) will throw a syntax error
print ("tuple3: ", tuple3, len(tuple3))

print ()

# tuple operations: limited since unlike lists, tuples are immutable

tuple4 = (1, 2, 3)

# you can get length of tuples just like lists
print ("tuple4: ", tuple4, len(tuple4))

# you can index tuples just like lists
print ("tuple4: ", tuple4[0], tuple4[1], tuple4[2])

# you can slice tuples just like lists
print ("tuple4[0:2]: ", tuple4[0:2])

print ()

# you can also nest tuples like lists
tuple5 = ((1, 2, 3), (4, 5, 6))
print ("tuple5: ", tuple5, len(tuple5))
print ("tuple5[1][2]: ", tuple5[1][2])

print ()

# you can convert lists to tuples
tuple6 = tuple ([1, 2, 3])
print ("tuple6: ", tuple6, len(tuple6))
```

```
print ()

# tuple packing and unpacking
# multiple assignment is essentially packing/unpacking in action

# tuple packing
tuple7 = 1, 2, 3
print ("tuple7: ", tuple7, len(tuple7))

# tuple unpacking
x, y, z = tuple7
print ("unpacked tuple7: ", x, y, z)

# Note that it is actually the comma which makes a tuple, not the parentheses.
# The parentheses are optional, except in the empty tuple case, or when they
# are needed to avoid syntactic ambiguity.
# For example, f(a, b, c) is a function call with three arguments,
# while f((a, b, c)) is a function call with a 3-tuple as the sole argument.

# test

tup1 = (2)
print(len(tup1))

tup2 = (1, 2, 3)
tup2[1] = 10

x, y, z = [10, 'Dog', 30]
print (x, y, z)

print ()
```