```
# -*- coding: utf-8 -*-
"""
Created on Tue Oct  6 10:15:40 2015
@author: hina
Reference: https://docs.python.org/3/tutorial/index.html
"""

print ()

# The while statements will evaluate till the while condition holds true

# let's use a while loop to print out the Fibonacci Series:
#    0, 1, 1, 2, 3, 5, 8, 13, 21, ...
#    so essentially, we start with the elements 0 and 1, and then each
#    successive element of the series is the sum of the previous two elements

print ("Fibonacci series using while loop: ")

a, b = 0, 1

while (b < 100):
    print (b, end = ', ')
    a, b = b, a+b
print ()

print ()

# note: Python uses indentation to identify code blocks
# so the following would produce the error:
#     "unindent does not match any outer indentation leve"

print ("Fibonacci series using while loop: ")

a, b = 0, 1

while (b < 100):
  print (b, end = ', ')
 a, b = b, a+b
print ()

print ()

# The "if" statement first evaluates the "if" condition.
#   - if the condition evaluates to true, it executes the corresponding code
#     block and stops
#   - if the condition evaluates to false, it evaluates the next-in-line
#     "elif" condition, and if that condition evaluates to true, it executes
#     the corresponding code block and stops
#   - if none of the "if" and "elif" conditions are true, it will default to
#     the execution of the code block corresponding to the "else" statement
#     (if one exists) and stop

word = 'Python'
# word = 'Jython'
# word = 'Anaconda'
# word = 'CIS415'
```

```
if (word.startswith('P')):
    print ("Word starts with P")
elif (word.endswith('n')):
    print ("Word ends with n")
elif (word.startswith('A')):
    print ("Word starts with A")
else:
    print ("Unknown word")

print ()

# The for statement in Python differs a bit from what you may be used to.
# Rather than always iterating over an arithmetic progression of numbers, or
# giving you the ability to define both the iteration step and halting
# condition, Python's for statement iterates over the items of any iterable
# sequence (such as a list, tuple, or string), in the order that they appear
# in the sequence

words = ['cat', 'dog', 'cow', 'parrot', 'hamster', 'goat']
for w in words:
    print (w, len(w))

print ()

# The range function generates arithmetic prgogressions

# range(n) -> 0,...,n-1 in increments of 1
for i in range(5):
    print (i, end=',')
print ()

# range(m,n) -> m,...,n-1 in increments of 1
# if m >= n, returns nothing
for i in range(3,10):
    print (i, end=',')
print ()

# range(m,n,k) -> m,...,<=n-1 in increments of k

# counts in positive increments if n > m, and k is positive
for i in range(3,10,2):
    print (i, end=',')
print ()

# counts in negative increments if n < m, and k is negative
for i in range(10,-30,-5):
    print (i, end=',')
print ()

words = ['jane', 'john', 'mark', 'harry', 'mike', 'ed']
wordlist = []
for i in range(len(words)):
    wordlist.append([i, words[i]])
print (wordlist)
print ()
```

```
print ()

# Note: The object returned by range() behaves as if it is a list,
# but in fact it isn't. It is an object which returns the successive items of
# the desired sequence when you iterate over it, but it doesn't really make
# the list, thus saving space.

print ("range is not a list: ", range(10))
print ("generate list underlying range: ", list(range(10)))

print ()

# break -
# breaks out of the smallest enclosing for or while loop.
for n in range(1, 10):
    if n % 2 == 0:
        print("found even number: ", n)
        break

print ()

# else -
# executed when the loop terminates thro exhaustion of the list (with for),
# or when the condition becomes false (with while),
# but not when the loop is terminated by a break statement

for n in range(1, 10, 2):
    if n % 2 == 0:
        print("found even number: ", n)
        break
else:
    # loop fell through without finding a factor
    print("even number not found")

print()

# continue -
# continues with the next iteration of the loop
for n in range(1, 10):
    if n % 2 == 0:
        print("even number: ", n)
        continue
    print("not an even number", n)

print ()

# Pass statements
#     the pass statement does nothing. It can be used when a statement is
#     required syntactically but the program requires no action.

#while True:
#    pass  # Busy-wait for keyboard interrupt (Ctrl+C)

#class MyEmptyClass:
#    pass  # create a minimal class
```

```
#def initlog(*args):
#    pass  # placehodler for function what implementing new code

# test

for i in range(1, -10, 2):
    print (i, end = ',')

while (True):
    pass

n = 1
while (n < 10):
 print (n)
   n = n + 1

if (5 < 3):
    print ("condition is true")
```