Control Flow Statements in Python

Shouke Wei, Ph.D. Professor

Email: shouke.wei@gmail.com

Objective

- If Conditional Statement
- While Loop
- For Loop

1. Conditional Statement

- `if…elif…else` are conditional statements, which
- provide you with the particular conditions to execute code
- helps automate the decision making process

1.1 If condition

```
if condition:
  expression
```

- the simplest form to make a decision based on whether the condition is true or not

In [3]:
```
n = 296
if n % 2 == 0: # true
  print('n is an even number')
```

```
n is an even number
```

1.2 if-else condition

```
if condition:
  expression 1
else:
  expression 2
```

- adds an additional step in the decision-making process
- The beginning of an `if-else` statement operates similar to a simple `if` statement; however,

---

- if the condition is false, instead of printing nothing, the indented expression under `else` will be printed
- An example:

In [ ]:
```
n = 287
if n % 2 = 0: # true
  print('n is a even number') # true
else:  # not true
  print('n is a odd number')
```

1.3 if-elif-else condition

- The most complex of these three conditions
- there are several conditions

```
if condition:
expression 1
elif condition:
expression 2
.
.
.
else:
express n
```

- you can place as many elif conditions as necessary between the if condition and the else condition ```

In [ ]:
```
saleFruit = ['Apple','Orange','Melon','Grape']
stockFruit = ['Apple','Orange','Melon','Grape']

# check if banana is on sale or in storehouse
if 'Banana' in saleFruit:
  print('Banana is selling.')
elif 'Banana' in stockFruit:
  print('Banana is in the storehouse')
else:
  print('Banana is out of stock!')
```

2. The While Loops

2.1 while loop

```
while <condition>:
    <statement(s)>
```

- Execute a set of statements as long as a condition is true, e.g.

In [5]:
```
num = 10

while num > 0:
  num -= 1  # num = num -1
  print(num)
```

```
9
8
7
6
5
4
3
2
1
0
```

2.2 The while-else loops

```
while <condition>:
    <statement(s)>
else:
    <additional_statement(s)>
```

- The `else` statement will run a block of code once the condition is no longer true, e.g.

In [ ]:
```
n = 1
while n < 7:
  n += 1 # n = n + 1
  print(n)
else:
  print("n is no longer less than 7")
```

2.3 The while-if-else loops

In [6]:
```
# guess the number

word = ' '
while word != 'big':
    word = input('Please input a word with the first letter of b: ')
    if word == 'big':
        print('Great! You got it.')
    else:
        print('Sorry. It is not correct. Please guess it again.')
```

```
Please input a word with the first letter of b: yes
Sorry. It is not correct. Please guess it again.
Please input a word with the first letter of b: but
Sorry. It is not correct. Please guess it again.
Please input a word with the first letter of b: big
Great! You got it.
```

2.4 The break statement

- The break statement will stop the loop even if the while condition is still true, e.g.

In [1]:
```
n = 1

while n < 7:
    n +=1
    print(n)

    if n == 4:
        break
```

```
2
3
4
```

2.5 The Continue Statement

- The `continue` statement we can stop the current iteration, and continue with the next

In [2]:
```
n = 1

while n < 7:
    n +=1

    if n == 4:
       continue
    print(n)
```

```
2
3
5
6
7
```

3. For Loops

3.1 For Loops

- A for loop is used for iterating over a sequence (a list, a tuple, a dictionary, a set, or a string)

```
for iterating_var in sequence:
 statements
```

In [7]:
```
fruitList = ['Apple','Cherry','Orange','Melon','Banana', 'Grape']

for items in fruitList:
  print(items)
```

```
Apple
Cherry
Orange
Melon
Banana
Grape
```

3.2 The break Statement

- The break statement stops the loop before it would finish looping through all the items

In [8]:
```
fruitList = ['Apple','Banana','Cherry','Orange','Melon', 'Grape']

for items in fruitList:
  print(items)

  if items == 'Orange':
    break
```

```
Apple
Banana
Cherry
Orange
```

- Exit the loop when item is "Orange", which includes Orange
- The following example also exit the loop when item is "Orange", but "Orange" is not printed

In [9]:
```
fruitList = ['Apple','Banana','Cherry','Orange','Melon', 'Grape']

for items in fruitList:

  if items == 'Orange':
    break
  print(items)
```

```
Apple
Banana
Cherry
```

3.3 The Continue Statement

- The `continue` statement stops the current iteration of the loop, and continue with the next
- It works as skiping an item

In [10]:
```
fruitList = ['Apple','Banana','Cherry','Orange','Melon', 'Grape']

for items in fruitList:
  if items == 'Orange':
    continue
  print(items)
```

```
Apple
Banana
Cherry
Melon
Grape
```

3.4 range() function

- returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number

In [12]:
```
range(6) # not the values of 0 to 6, but the values 0 to 5

for n in range(6):
  print(n)
```

```
0
1
2
3
4
5
```

- it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

In [13]:
```
for n in range(2, 6):
  print(n)
```

```
2
3
4
5
```

- it is possible to specify the increment value by adding a third parameter: range(2, 30, 3):

In [14]:
```
for n in range(2,10,2):
  print(n)
```

```
2
4
6
8
```

3.5 Else in for loops

- The else keyword in a for loop specifies a block of code to be executed when the loop is finished

In [15]:
```
fruitList = ['Apple','Banana','Cherry','Orange','Melon', 'Grape']

for items in fruitList:
  print(items)

else:
  print("This is the end!")
```

```
Apple
Banana
Cherry
Orange
Melon
Grape
This is the end!
```

- If the loop breaks, the else block is not executed.

In [16]:
```
fruitList = ['Apple','Banana','Cherry','Orange','Melon', 'Grape']

for items in fruitList:

  if items == 'Orange':
    break
  print(items)

else:
  print("This is the end!")
```

```
Apple
Banana
Cherry
```

3.6 Nested Loops

- A nested loop is a loop inside a loop
- The "inner loop" will be executed one time for each iteration of the "outer loop"

In [19]:
```
attributeList = ['Red','Big','Sweet']
fruitList = ['Apple','Banana','Cherry']

for x in attributeList:
  for y in fruitList:
    print(x, y)
```

```
Red Apple
Red Banana
Red Cherry
Big Apple
Big Banana
Big Cherry
Sweet Apple
Sweet Banana
Sweet Cherry
```