# The power of *if* statement

In this lecture, we will learn the conditional statement *if*,
It is a conditional statement that test for some condition, and execute some codes accordingly.
Let's see some examples.

**Created by:** John C.S. Lui on May 25, 2018.

**Important note:** *If you want to use and modify this notebook file, please acknowledge the author.*

In [ ]:
```
# Create a list of CS subjects
cs_subjects = ['networks', 'machine learning', 'ai', 'programming langu
favorite_subject = 'machine learning'

for subject in cs_subjects:
    if subject == favorite_subject:        # condition if.   Note that
        print('My favorite subject is:', subject.title())
    else:                                  # Takes the 'false' conditi
        print('I hate %s' %subject.title())
```

In [ ]:
```
# Create a list of CS subjects
cs_subjects = ['networks', 'machine learning', 'ai', 'programming langu
favorite_subject = ['machine learning', 'networks', 'theory']   # chan

for subject in cs_subjects:
    if subject in favorite_subject:        # condition if.   Check whe
        print('My favorite subject is:', subject.title())
    else:                                  # Takes the 'false' conditi
        print('I hate %s' %subject.title())
```

# Logical comparison operators

In Python, we have the following:

- equality (==)
- inequlaity (!=)
- greater than (>)
- greater than or equal to (>=)
- less than (<)
- less than or equal to (<=)
- and operator (and)
- or operator (or)

In [ ]:
```
# Let's try some of this logical comparison and print out their return
print ("1.", 3==3)
print ("2.", 5==8)
print ("3.", 1==1.0)
print ("4.", 'John'=='John')
print ("5.", 'John'=='john')
print ("6.", 'John'.lower() =='john')
print ("7.", '4' == str(4))
```

In [ ]:
```
# Let's try the ">=" operator
print ("1.", 10 >= 8)
print ("2.", 10 >= 80)
print ("3.", 10 >= 10)
```

In [ ]:
```
# Let's try "<" operator
print ("1.", 10 < 8)
print ("2.", 1  < 3.0)
```

In [ ]:
```
# Let's try "<=" operator
print ("1.", 10 <= 8)
print ("2.", 1  <= 3.0)
```

In [ ]:
```
print("1.", 8 <= 10 and 5 < 10)
print("2.", 8 <= 10 and 5 > 10)
print("3.", 10 <= 8 or 10 > 5)
```

# Let's check whether an item is in the list

In [ ]:
```
# Define a list
list_number = [0, 1, 3, 'john', 4, 'peter', 'paul']

if 'john' in list_number:
    print ('It is in the list')
else:
    print ('Not in the list')
```

## Simple *if* statement

In [ ]:
```
list_number = [0, 1, 3, 'john', 4, 'peter', 'paul']
if len(list_number) >= 3:
    print('The length of "list_number" is at least 3')
```

# Simpe *if-else* statement

In [ ]:
```
list_number = [0, 1, 3, 'john', 4, 'peter', 'paul']
if len(list_number) <= 3:
    print('The length of "list_number" is less than or equal to 3')
else:
    print('The length of "list_number" is greater than 3')
```

## The *if-elif .... else* chain

This is useful if you want to test *a series* of conditions.
In an *if-elif-else* chain, once a test passes, the rest of the conditions are ignored.

In [ ]:
```
# Define a list
list_number = [0, 1, 3, 'john', 4, 'peter', 'paul']

if len(list_number) == 0:
    print('There is no item in the list')
elif len(list_number) <= 3:
    print('There is at least one and at most 3 items in the list.')
elif len(list_number) <= 7:
    print('There is at least 4 and at most 7 items in the list')
else:
    print('There is greater than 7 items in the list')
```

## Question

What is the difference between *if-elif* and a series of many *if* ?

## Exercise

Write a program to first prompt the user for an input (hint: look up the function *input()* ) for a day.
Then create a list with all 7 days (assume you only use lower case).
Use *if... elif* to test whether the input day is a weekday or weekend.
Print out whether the input is a weekeday or weekend or an input error.

## Going through a series of test

What if you want to run a series of test? Let's try.

In [ ]:
```
# define a function
def print_message (subject, flag):
    if flag==True:
        print("For " + subject.title() + ", it is John's favorite subj
    else:
        print("For " + subject.title() + ", it is definitely not John'

# define a list
cs_subjects = ['networks', 'machine learning', 'ai', 'programming langu

for subject in cs_subjects:
    if subject == 'networks':
        print_message(subject, True)
    if subject == 'machine learning':
        print_message(subject, True)
    if subject == 'ai':
        print_message(subject, False)
    if subject == 'programming languages':
        print_message(subject, False)
    if subject == 'distributed systems':
        print_message(subject, True)
    if subject == 'theory':
        print_message(subject, True)
```

In [ ]:
```
## Let's see how we can write a simply code
# define a function
def print_message (subject, flag):
    if flag==True:
        print("For " + subject.title() + ", it is John's favorite subj
    else:
        print("For " + subject.title() + ", it is definitely not John'

# define a list
cs_subjects = ['networks', 'machine learning', 'ai', 'programming langu

favorite_subjects = ['networks', 'machine learning', 'distributed syste

for subject in cs_subjects:
    if subject in favorite_subjects:
        print_message(subject, True)
    else:
        print_message(subject, False)
```

# True and False values

In general, any **non-zero** or **non-empty** value will be evaluated as True. Let check.

In [ ]:
```
# Define a list
number_list1 = [-3, -2, -1, 0, 1, 2, 3]
number_list2 = [-3.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0]

# Let's test
for number in number_list1:
    if number:
        print("For", number, "it is True.")
    else:
        print("For", number, "it is False.")
print ("----------------")

# Let's test
for number in number_list2:
    if number:
        print("For", number, "it is True.")
    else:
        print("For", number, "it is False.")

print ("----------------")

if '':
    print ("*** It is True.")
else:
    print ("*** It is False")

print ("----------------")

if ' ':   # This is a space, so it is non-empyt
    print ("*** It is True.")
else:
    print ("*** It is False")

print ("----------------")

if None:   # None is a special object in Python. It evaluates to False
    print ("*** It is True.")
else:
    print ("*** It is False.")
```

In [ ]:
```

```