py4kids (https://github.com/wgong/py4kids)

Control Flow and Loop

In this lesson, we learn :

- What is Boolean Type
- How to use condition to control flow
- How to loop
- How to program

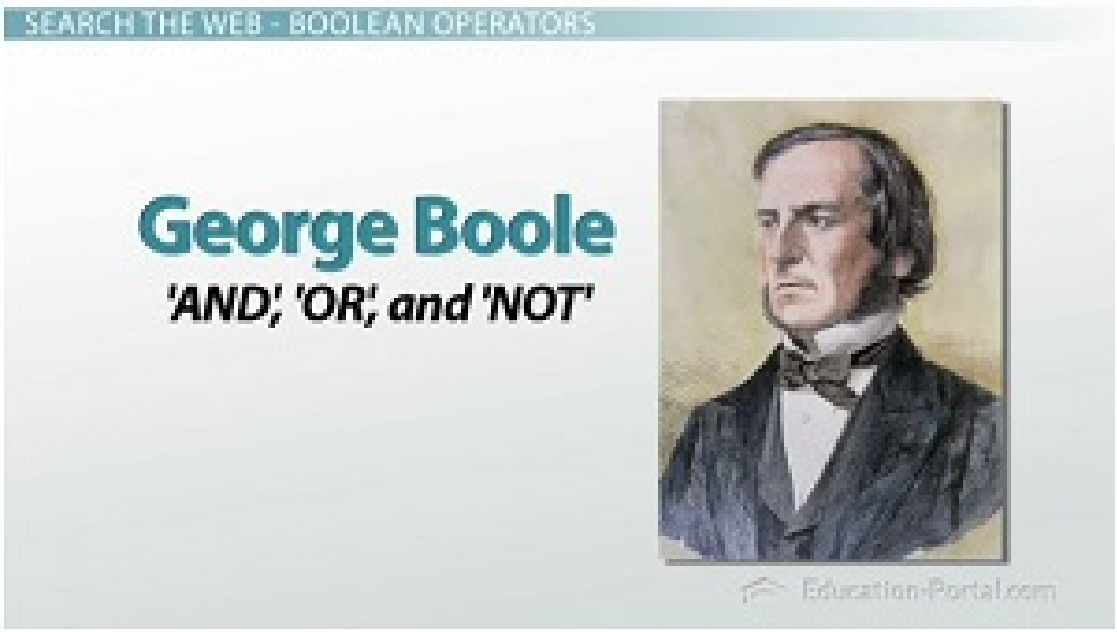
In [1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

Out[1]:

Boolean type: True or False

The Python type for storing true/false (or yes/no) values is called bool, named after the British mathematician, George Boole.

George Boole (https://www.wikiwand.com/en/George_Boole) created Boolean Algebra, which is the basis of all modern computer arithmetic, he also made great contributions to Differential Equations and Theory of Probability.



In [2]:
```
q1 = "Rabbit runs faster than turtle. True or False?"
a1 = True
a11 = "True"
# unless rabbit sleeps
```

In [3]:
```
type(a1), type(a11)
```

Out[3]:
```
(bool, str)
```

In [4]:
```
# 电脑会思维吗?
q2 = "Computer can think. True or False?"
a2 = False
type(a2)
```

Out[4]:
```
bool
```

In [5]:
```
print(a2)
```

```
False
```

bool type has only two values: [True, False]

Special cases

In python, None is a special value. It means NOTHING, null, nill.

In [6]:
```
bool(None), bool(''), bool("None")
```

Out[6]:
```
(False, False, True)
```

1 is Ture, 0 is False

In [7]:
```
bool(0), bool(1), bool(0.1)
```

Out[7]:
```
(False, True, True)
```

comparison creates condition

Table 5-1: Symbols for Conditions

Symbol	Definition
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

In [8]:
```
a = 100; b = 100
condition1 = (a == b)
type(condition1)
```

Out[8]:
```
bool
```

In [9]:
```
print(condition1)
```

```
True
```

In [10]:
```
c2 = a > b
c3 = a < b
c4 = (a != b)
c5 = (a >= b)
c6 = (a <= b)
```

In [11]:
```
print(c2, c3, c4, c5, c6)
```

```
False False False True True
```

"False" is True

In [12]:
```
s1 = "False"
print(s1)
```

```
False
```

In [13]:
```
b1 = False
print(b1)
```

```
False
```

In [14]:
```
type(s1), type(b1)
```

Out[14]: (str, bool)

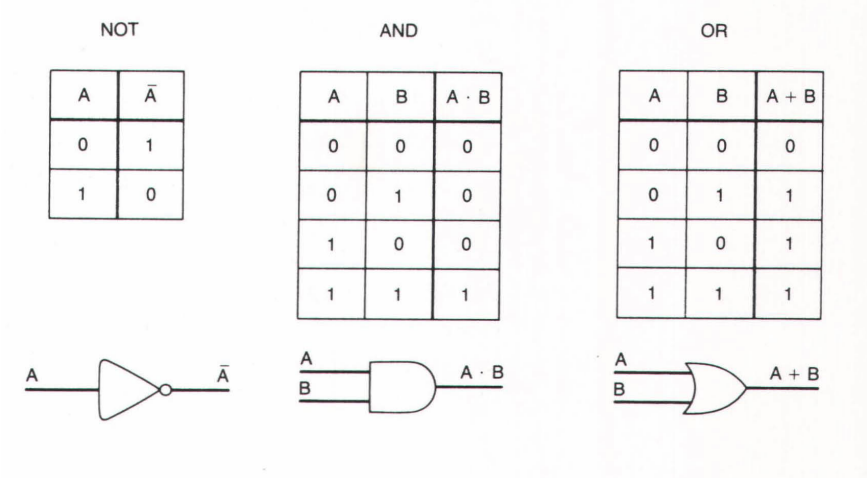
In [15]:
```
bool(s1)
```

Out[15]: True

In [16]:
```
bool(False)
```

Out[16]: False

logic operation: and, or, not



In [17]:
```
x = 1
y = 22
z = 333
c7 = (x < y)
c8 = (y < z)
c9 = c7 and c8
c10 = c7 or c8
c11 = not c7
print(c7, c8, c9, c10, c11)
```

```
True True True True False
```

data-structure summary

below is a list of basic python data structures we have learned

Data Type	Type Name	Example
Boolean	bool	True, False
Integer	int	1, 100, 123
Real number	float	3.14159, 1.0e3
Text or String	str	'Hello', "你好"
List	list	[1,2,3, "A", "B", "C"]
Tuple	tuple	(100,200,300)
Set	set	{100,200,300}
Dictionary, Map, Table	dict	{1001: "John", 1002: "Jane"}

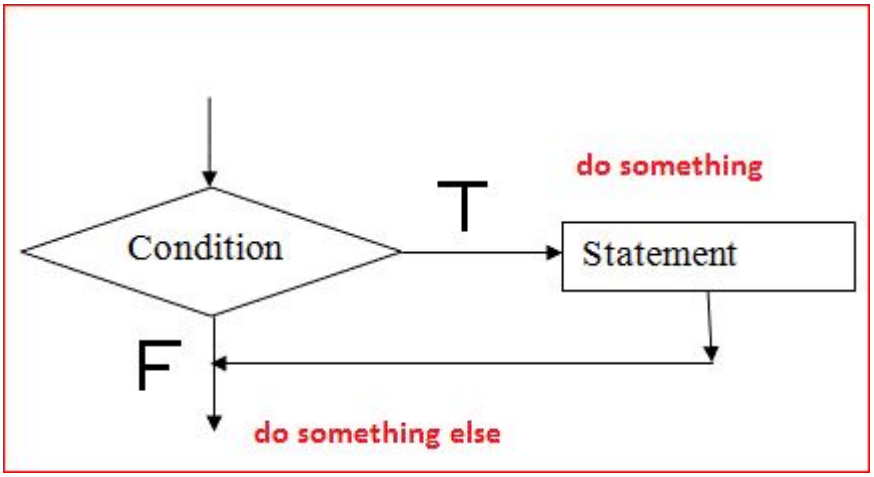
Later on, we will learn how to design our own data structure using class/object

In [18]:
```
a = {1001: "John", 1002: "Jane"}
type(a)
```

Out[18]:
```
dict
```

If-elif-else

Condition ==> Decision-making ==> Action

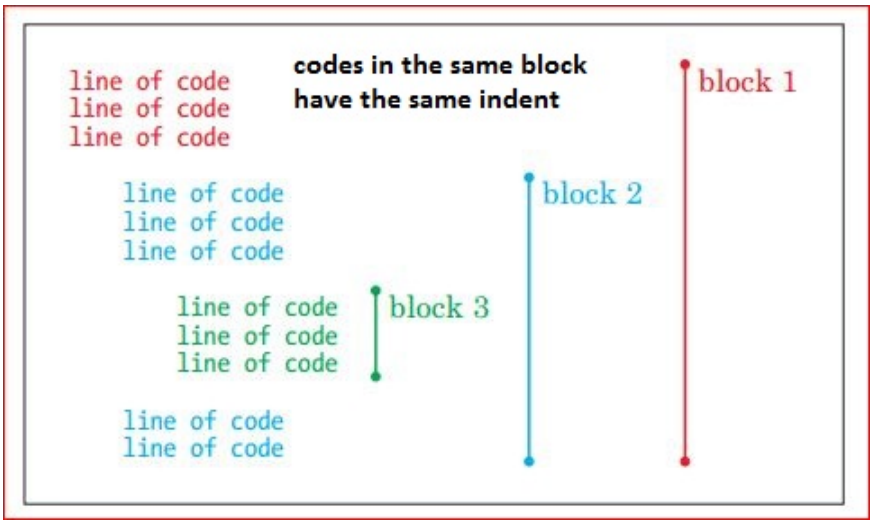


Note: read Chapter 5 of "Python for Kids" textbook which has a very good explanation

if statement

Indent is unique and critical feature in python

Many people uses 4 whitespaces for indent, but no fixed rule. 2 or 8 spaces are ok. The key is to make your codes humanly readable



In [19]:
```
your_age = 12
if your_age < 13:
    print('Your age is ', your_age)
    print('You are not a teenager yet')
```

```
Your age is  12
You are not a teenager yet
```

if-else statement

In [20]:
```
your_age = 13
if your_age < 13:
    print('Your age is ', your_age)
    print('You are a baby')
else:
    print('Your age is ', your_age)
    print('You are a teenager')
```

```
Your age is  13
You are a teenager
```

condition in else branch is implied, i.e., not if condition

if-elif-else statement

In [21]:
```
your_age = 21
if your_age < 13:
    print('Your age is ', your_age)
    print('You are a baby')
elif your_age < 18:
    print('Your age is ', your_age)
    print('You are a teenager')
else:
    print('Your age is ', your_age)
    print('You are an adult')
```

```
Your age is  21
You are an adult
```

nested if-else statement

In [22]:
```
your_age = 21
gender = "Female"

if your_age < 13:
    print('Your age is ', your_age)

    if gender == 'Male':
        print('You are a baby boy')
    else:
        print('You are a baby girl')

elif your_age < 18:
    print('Your age is ', your_age)
    if gender == 'Male':
        print('You are a teenager boy')
    else:
        print('You are a teenager girl')
else:
    print('Your age is ', your_age)
    if gender == 'Male':
        print('You are a man')
    else:
        print('You are a woman')
```

```
Your age is  21
You are a woman
```

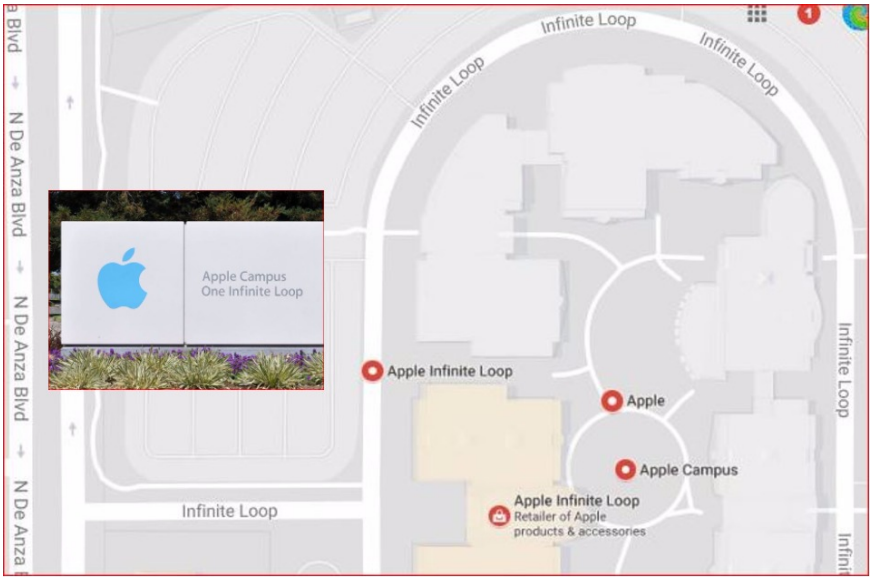
Loop

Examples

A few examples of looping

- apple headquarter
- computer operating system

- smart phone
- event loop for game, GUI
- web site
- our daily-routine
- Moon orbiting Earth, Earth around Sun, Sun moving in Milky Way
- four seasons



for loop

- **for** is a keyword for looping,
- repeat within the loop
- usually iteration over a finite set

loop over a list

In [23]:
```
colors = ['red','green','blue','yellow','black','white']
```

In [24]:
```
for c in colors:
    print("color is ", c)
```

```
color is  red
color is  green
color is  blue
color is  yellow
color is  black
color is  white
```

range() - a useful function to build a number list

In [25]:
```
number_list = range(10)
```

In [26]:
```
print(number_list)
```

```
range(0, 10)
```

In [27]:
```
for n in number_list:
    print("n= ", n)
```

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

In [28]:
```
number_list_2 = range(10, 100, 20)
# 10 is starting number
# 100 is the ending number
# 20 is the stride (or step)

for n in number_list_2:
    print("n= ", n)
```

```
n=  10
n=  30
n=  50
n=  70
n=  90
```

loop over a string

In [29]:
```
make_a_sentence = "python string can be as small as a character, or as large as a
```

In [30]:
```
for char in make_a_sentence:
    print(char)
```

```
p
y
t
h
o
n

s
t
r
i
n
g

c
a
n

b
e

a
s

s
m
a
l
l

a
s

a

c
h
a
r
a
c
t
e
r
,

o
r

a
s

l
a
```

```
r
g
e

a
s

a

l
i
b
r
a
r
y
.
```

loop over a set

In [31]:
```
shapes = {'circle', 'square', 'triangle', 'pentagon'}
```

In [32]:
```
type(shapes)
```

Out[32]:
```
set
```

In [33]:
```
for s in shapes:
    print(s)
```

```
circle
pentagon
square
triangle
```

loop over a dictionary

In [34]:
```
dict1 = {'China': '中国',
  'England': '英国',
  'France': '法国',
  'Germany': '德国',
  'India': '印度',
  'Japan': '日本',
  'USA': '美国'}
```

In [35]:
```
for item in dict1:
    print(item)
```

```
England
France
India
USA
China
Japan
Germany
```

In [36]:
```
print(dict1.items())
```

```
dict_items([('England', '英国'), ('France', '法国'), ('India', '印度'), ('USA',
 '美国'), ('China', '中国'), ('Japan', '日本'), ('Germany', '德国')])
```

In [37]:
```
for key,value in dict1.items():
    print('key=', key,' \t: ', 'value=',value)
```

```
key= England    :  value= 英国
key= France     :  value= 法国
key= India      :  value= 印度
key= USA        :  value= 美国
key= China      :  value= 中国
key= Japan      :  value= 日本
key= Germany    :  value= 德国
```

In [38]:
```
# how to track loop - use a counter

# initialize the counter before loop starts
n = 0
for item in dict1:
    n = n + 1  # increment counter by 1
    print('loop counter = %d' % n)
    print('\t\tkey=', item)
```

```
loop counter = 1
                key= England
loop counter = 2
                key= France
loop counter = 3
                key= India
loop counter = 4
                key= USA
loop counter = 5
                key= China
loop counter = 6
                key= Japan
loop counter = 7
                key= Germany
```

In [39]:
```
# how to loop thru a dictionary

# initialize the counter before loop starts
n = 0
for item in dict1:
    n = n + 1  # increment counter by 1
    print('loop counter = %d' % n)
    print('\t\tKey  =', item)
    print('\t\tValue=', dict1[item])
```

```
loop counter = 1
                Key  = England
                Value= 英国
loop counter = 2
                Key  = France
                Value= 法国
loop counter = 3
                Key  = India
                Value= 印度
loop counter = 4
                Key  = USA
                Value= 美国
loop counter = 5
                Key  = China
                Value= 中国
loop counter = 6
                Key  = Japan
                Value= 日本
loop counter = 7
                Key  = Germany
                Value= 德国
```

while loop

- usually for an infinite loop

how to create an infinite loop

```
while True:
    print("looping forever")
```

In [40]:
```
# list even number less than 20
n1 = 0
while n1 < 20:
    if n1 % 2 == 0:
        print(n1, ' is an even number')
    else:
        pass       # pass means pass, does nothing
    n1 = n1+1
```

```
0  is an even number
2  is an even number
4  is an even number
6  is an even number
8  is an even number
10  is an even number
12  is an even number
14  is an even number
16  is an even number
18  is an even number
```

how to control an infinite loop

- break
- continue

random number generator

In [41]:
```
from random import randint
# pick a number between 0 and 1000 randomly
a_random_int = randint(0,1000)
print('You picked number: ', a_random_int)
```

```
You picked number:  531
```

In [42]:
```
while True:  # this is an infinite loop
    your_number_pick = randint(0,1000)
    print(your_number_pick)
    if your_number_pick % 9 == 0:
        print('\tCongratulation!  you picked a 9-multiple')
        break
```

```
272
844
202
364
197
187
697
130
920
746
968
608
210
28
914
951
128
971
633
542
413
53
49
902
986
995
671
318
47
378
        Congratulation!  you picked a 9-multiple
```

a little lottory app

In [43]:
```
my_lucky_number = 7
l_counter = 0
while True:           # this is an infinite loop
    your_number_pick = randint(0,1000)
    l_counter = l_counter + 1
    print("[%03d] %5d" % (l_counter,your_number_pick))
    if your_number_pick % my_lucky_number == 0:
        print('\n*** Good Luck ***  You got a %d-multiple' % my_lucky_number)
        break         # will terminate the infinite loop
    else:
        continue      # go on forever
```

```
[001]   531
[002]   589
[003]   479
[004]   707

*** Good Luck ***  You got a 7-multiple
```

In [ ]:

In [ ]: