Introduction to Python: Loops and Input Functions

Here, we will explore other looping construct under Python, for example,
we have **while**, and we also illustrate how to allow user to input arguments into our program.

Created by: John C.S. Lui on May 28, 2018.

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

While syntax

Initialize the condition

flag == True

The while loop

while flag: # execute the following statements until the flag becomes False
    # statement 1
    # statement 2
    # .....
    # statement n

- All **while** loop has an initial condition as true.
- The **while** statement includes a condition to test.
- All statements in the loop will be executed as long as the condition is true.
- When the condition changes to False (within the statement loop), the loop stops
- Codes that are defined after the **while** loop will be executed

Let's illustrate it with some examples.

In [ ]:
```
# Computing the sum from 1 to upper_limit

upper_limit = 10   # Define the upper limit

sum = 0
num = 1

while num <= upper_limit :
    sum = sum + num
    num = num + 1

print("The sum from 1 to", upper_limit, "is", sum)
```

Example

A Fibonacci series is a series of numbers in which each number is the sum of the two preceding numbers.
For example, we have the following series 1, 1, 2, 3, 5, 8, etc.
Use the **while** loop to generate a series for up to *upper_limit* numbers.

In [ ]:
```
upper_limit = 10   # set upper limit for the Fibonacci series
previous_num = 0
current_num = 1
next_num  = 0
index = 1

print("The Fibonacci series: " + str(1) +",", end=" ")  # instead of n
while index <= upper_limit:
    next_num = previous_num + current_num          # generate next num
    if index < upper_limit:                        # if not last numbe
        print (str(next_num) +",", end=" ")
    else:                                          # if last number, g
        print (str(next_num) + ".")
    previous_num = current_num                     # change previous a
    current_num = next_num
    index = index + 1                              # modifying the con
```

Exercise

Use the **while** loop to generate the first 10 numbers of 2^n, where n is an integer with an upper bound.

Allowing user input

We have learnt how to generate output, but how do we accept input by the users of the program?
It is by the *input()* function. Let's illustrate.

General syntax for input

variable = input ('Message you want to show: ")

Example

In [ ]:
```
# create a list of teachers in the first floor
teachers = ['john c. s. lui', 'patrick p. c. lee', 'james cheng', 'wei

# Ask the user to input a name
name = input('Enter a name of the teacher:')

# Now check whether the name is in the list

if name in teachers:
    print("Yes, ", name.title(), "is in the list")
else:
    print("Sorry, ", name.title(), "is NOT in the list")
```

Note on *input()* and *raw_input()* in Python 2.7

In Python 3, you use *input()* to get a **string** back. In Python 2.7, the counter-part is *raw_input()*

Exercise

Write a program with the following:

- Create a list of your favorite sports
- Print out your favorite sports
- Prompt user to enter another sport
- If the sport is not in the list, append it to the list, else print out an error message
- display the content of the current list

Examples

In [ ]:
```
# Keep adding new names into the list
teachers = []   # create an empty list

name = ''       # initialize the name

while name.upper() != 'QUIT':
    name = input ('Input the name of a professor whose office is on th
    if name.upper() != 'QUIT':
        teachers.append(name)
        print("Right now the list is:", teachers)

print("Done!")
```

In [ ]:
```
# Give the user some context for "menu input".
print("\nWelcome to the nature center. What would you like to do?")

# Set an initial value for choice other than the value for 'quit'.
choice = ''

# Start a loop that runs until the user enters the value for 'quit'.
while choice != 'q':
    # Give all the choices in a series of print statements.
    print("\n[1] Enter 1 to take a course in Python.")
    print("[2] Enter 2 to take a course in C++.")
    print("[3] Enter 3 to take a course in Java.")
    print("[q] Enter q to quit.")

    # Ask for the user's choice.
    choice = input("\nWhich course you like to take ? ")

    # Respond to the user's choice.
    if choice == '1':
        print("\nPython is great. Have fun!\n")
    elif choice == '2':
        print("\nC++ is also a good language. It has good performance!
    elif choice == '3':
        print("\nYou are to take Java? Are you sure?\n")
    elif choice == 'q':
        print("\nOK, I know you have enough programming languages\n")
    else:
        print("\nI don't understand your choice, please try again.\n")

# Print a message that we are all finished.
print("Thanks again, Chiao.")
```

In [ ]:
```
# Example of using while loop to go through a list

# Start with a list of teachers, and an empty list of smart teachers.
teachers = ['john c. s. lui', 'patrick p. c. lee', 'james cheng', 'wei
smart_teachers = []

# Let's go through the teachers
while len(teachers) != 1:  # process all but the first item in the tea

    # Get the last teachers from the list, put he/she in the smart_tea
    current_teacher = teachers.pop()
    print("We have %s from the list!" % current_teacher.title())

    # Move it to the smart_teachers list
    smart_teachers.append(current_teacher)

# Prove that we have finished confirming all users.
print("\nSmart teachers are: ")
for people in smart_teachers:
    print('- ' + people.title())

print("\nDumb teachers are:")
for people in teachers:
    print('- ' + people.title())
```

Final examples

In [ ]:
```
# A program with while
a, b = 0, 1          # parallel assignment
while b < 10:
    print(b)
    a, b = b, a+b
print ('end')
```

Nested while

In [ ]:
```
x = 0
while x < 3:   # outer loop
    y = 0
    while y < 3:   # inner loop
        print("x =", x, ";", "y =", y)
        y = y+1
    x = x+1
```

In [ ]:
```

```