CS1302 - Lecture 4 Iteration
(Dr Helena WONG)
==========

[Summary of Topic 4] Iteration

[Helena's supplements for introduction]

[Iteration.ipynb]

1 Motivation

2 Loops
- While Loop
- For Loop
- While Loop vs For Loop

3 Break / Continue / Else Constructs of a Loop
- Breaking out of a loop
- Continue to Next Iteration
- Else construct for a loop

==========

[Helena's supplements for introduction]

For-loop: Introductory examples

```
for item in 1,2,3,4: print(item, end=". ")     # 1. 2. 3. 4.

for item in range(1,5): print(item, end=". ")  # 1. 2. 3. 4.

w   for item in "Good": print(item, end=". ")      # G. o. o. d.
```

The for statement iterates over a sequence of elements (such as a string, *tuple* or *list*) or other iterable objects:
https://docs.python.org/3/reference/compound_stmts.html#for

Iterable (https://docs.python.org/3/glossary.html#term-iterable)

An object from which we can get members one at a time.
Can be used in a for-loop
E.g. str, range, list, tuple

Strings and **Ranges** are **iterable**
for item in range(1,5): print(item, end=". ") # 1. 2. 3. 4.
for item in "Good": print(item, end=". ") # G. o. o. d.

tuple and **list** (an important CS1302 topic to come)
What are tuple and list ?
They are sequences of items.

Example: Tuple asignment

```
# t1: A tuple of strings, t2: A tuple of tuples

t1 = "A+", "A", "A-", "B+"

t2 = (0, "A+"), (1, "A"), (2, "A-"), (3, "B+")
```

```
for g in t1:
    print(g, end=". ")
print()

for i, g in t2:
    print(i, ":", g, end=". ")
print()

# Output 2 lines: A+. A. A-. B+. and 0 : A+. 1 : A. 2 : A-. 3 : B+.

    # Note: programmers love to start indexing from zero
```

enumerate (https://docs.python.org/3/library/functions.html#enumerate)

.. converts an iterable sequence and returns an enumerate object (also an iterable),
.. adds a counter and returns it in a form of (index,element). Often used in for loops.

```
t3 = enumerate(t1)
for i, item in t3: print(i, ":", item, end=". ")    # 0 : A+. 1 : A. 2 : A-. 3 : B+.
```

What is a range? (https://docs.python.org/3/library/stdtypes.html#range)

.. sequence of numbers commonly used for looping a specific number of times in for loops.

c.f. how we iterate a string to get the characters, we can do so for a range

A string is an iterable

```
x = "hi!"

my_tuple = tuple(x)
print(my_tuple) # we have obtained a tuple of characters from x

my_list = list(x)
print(my_list) # we have obtained a list of characters from x
```

A range is an iterable

```
x = range(100, 103)
my_tuple = tuple(x)
print(my_tuple) # we have obtained a tuple of numbers from x

my_list = list(x)
print(my_list) # we have obtained a list of numbers from x
```

The [] notation

```
x = "hi!"
print(x[0], x[1], x[2])    # h i !
print(x[3]) # error

x = range(100,103)
print(x[0], x[1], x[2])    # 100 101 102
print(x[3]) # error
```

==========

==========
The lecture notebook [Iteration.ipynb]:
==========

1 Motivation

```
# 1) get an input integer num
# 2) Want to list the numbers from 1 to num

num = int(input("Give me the number:"))
if 1 <= num: print(1)
if 2 <= num: print(2)
if 3 <= num: print(3)
if 4 <= num: print(4)
if 5 <= num: print(5)
if 6 <= num: print(6)

# What if input is 200?  This approach is poor.
```

2 Loops

```
===============
       num = int(input("Give me the number:")) # assume 10
       num = num+1 # gets 11
       for x in range(1, num): # x: 1,2,3,...10 (11 not included)
           print(x)
===============

num = int(input("Give me the number:"))
i = 1
while i <= num:
    print(i)
    i += 1

===============
# alternative
i = -1
while i != num:
    i += 1
    print(i)
===============
# infinite loop:
i = 1
while i <= num:
    print(i)
    i + 1
===============
# infinite loop:
i = 1
while i <= num:
    print(i)
i + 1
```

2.1 **start**, **stop**, **step** of **a range**:

```
===============
for i in range(1, 10):
     print(i, end=" ")

===============
for i in range(0, 10, 2):
     print(i, end=" ")

===============
list(range(1, 10)) # creates a list
===============
list(range(0, 10, 2))
```

```
==============
list(range(-15, -1))
# is it the same as range(0,-15,-1): print 0,-1,-2,..-15?
# is it the same as range(-15,-1,1): print -15,-14,-13,..-2?
==============

# step by 0.5 until num
        # input 3
        # output 0.0, 0.5, ... until 3.0

        # num = 3
        # for i in range(0.0, num + 0.5, 0.5):
        #     print(i)

        # for i in range(0,7): # i:0,1,2,3,4,5,6 (7 excluded)
        #     print(i/2)

        num = 4
        for i in range(0,num*2+1): # i:0,1,2,3,4,5,6,7,8 (num*2+1 excluded)
             print(i/2)
```

2.2 Using the len function; Using [], the indexing operator

```
        message = 'loop'
        print(len(message))
        print(message[0], message[1], message[2], message[3])
        print(message[-1], message[-2], message[-3], message[-4])

        for i in range(len(message)):
            print(message[i], end=" ")

        for c in message:
            print(c, end=" ")
==============
# reverse_print

        def reverse_print(message): # if "apple" length is 5
            # message[4], message[3], ... message[0]
            for i in range(len(message)-1, -1, -1): # i: 4,3,2,1,0 (-1 excluded)
                print(message[i], end=" ")

        reverse_print("apple")  # "elppa"

        ===============
        def reverse_print(message): # if "apple" length is 5
            # message[4], message[3], ... message[0]
            for i in range(0,len(message)): # i:0,1,2,3,..4 (5 excluded)
                print(message[len(message)-1-i], end=" ") # 5-1-i: 4,3,2,1,0

        reverse_print("apple")

        ===============
        def reverse_print(message): # if "apple" length is 5
            # message[-1], message[-2], ... message[-5]
            for i in range(0,len(message)): # i:0,1,2,3,..4 (5 excluded)
                print(message[-i-1], end=" ")

        reverse_print("apple")

==============
```

2.3 For Loop vs While Loop

Definite vs Indefinite loop

- `for` is often used for a *definite loop* which has a definite number of iterations before execution.
- `while` is often used for an *indefinite loop* where the number of iterations is unknown before execution.

It is always possible to replace a `for` loop by a `while` loop.

```
sum = 0
i = 1

while i<=100:
    sum = sum + i
    i = i+1

print(sum) # 5050
print(i) # <================ what is it?

================
sum = 0

for i in range(1,101): # i already determined 0,1,2...100
    sum = sum + i

print(sum)
print(i) # <================ what is it?
```

```
================
# 1) get the sum for 1,2,3,.......
# 2) stop when the sum is over 80000

sum = 0
i = 1

while sum<=80000:
    sum = sum + i
    i = i+1

print(sum)

================
# Be careful when using while-loop:  infinite loop

num = 5        # num = int(input('>'))
i = 0
while i!=num+1: # i: 0, 1, 2, 3, 4, 5, (6: stop)
    print(i)
    i += 1
================
num = -5        # num = int(input('>'))
i = 0
while i!=num+1: # if we enter -5, num=-5, --->i!=-4, 0,1,2,3,4.....
    print(i)
    i += 1
================
num = 5        # num = int(input('>'))
for i in range(num+1):
    print(i)
================
num = -5        # num = int(input('>'))
for i in range(1, num + 1): # i: no values
    print(i, end=" ")

================
```

2.4 For Loop vs While Loop

Compare the speeds

We can compare the speed of the `for` loop and `while` loop using the cell magic %%timeit

```
%%timeit
for i in range(5): cur_count = i
```

```
%%timeit
for i in range(5): cur_count = i
```

```
113 ns ± 0.704 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

```
%%timeit
i = 0
while i < 5:
    cur_count = i
    i += 1
```

```
115 ns ± 0.31 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
```

```
%%timeit
for i in range(5000000): cur_count = i
```

```
65.2 ms ± 3.37 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
%%timeit
i = 0
while i < 5000000:
    cur_count = i
    i += 1
```

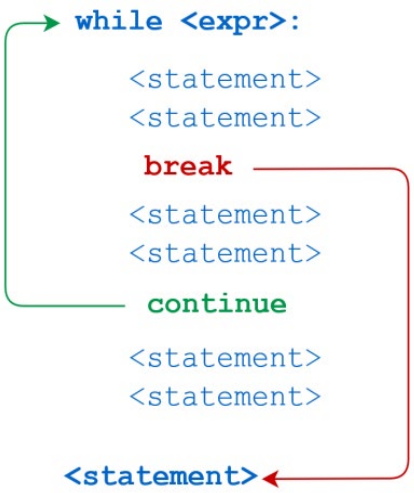
```
110 ms ± 8.24 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Atto a	10^{-18}	0.000 000 000 000 000 001
Femto f	10^{-15}	0.000 000 000 000 001
Pico p	10^{-12}	0.000 000 000 001
Nano n	10^{-9}	0.000 000 001
Micro μ	10^{-6}	0.000 001
Milli m	10^{-3}	0.001
Centi c	10^{-2}	0.01
Deci d	10^{-1}	0.1
	10^{0}	1
Deca da	10^{1}	10
Hecto h	10^{2}	100
Kilo k	10^{3}	1 000
Mega M	10^{6}	1 000 000
Giga G	10^{9}	1 000 000 000
Tera T	10^{12}	1 000 000 000 000

https://www.quora.com/What-is-the-difference-between-nanoseconds-microseconds-and-milliseconds

3 Break / Continue / Else Constructs of a Loop

```
#    break terminates the whole loop
#    but continue only terminates the current loop
```




```
===============
message = "string"
for c in message:
    if c=="i":
        break # <============ example of break
    print(c, end="")

print("\nThe end")

===============
while True:
    message = input("Input something please:")
    if message:
        break # <============ example of break

print("You entered:", message)

===============
message = "string"
for c in message:
    if c=="i":
        continue # <============ example of continue
    print(c, end="")

print("\nThe end")

===============
while True:
    message = input("Input something please:")
    if not message:
        continue # <============ example of continue
    print("You entered:", message)

# Try: Rewrite without continue

===============
```

```
        More examples
        ===============
                # 1) try the sum for 1,2,3,....... 10000000
                # 2) stop when the sum is over 800000

                        sum = 0

                        for i in range(10000000):
                            sum = sum + i
                            if sum>800000:
                                break # <============ example of break

                        print("1 to", i, "we get the sum: ", sum)

                ===============
                # 1) get the sum for 1,2,3,....... 10000000
                # 2) skip multiples of 11

                        sum = 0

                        for i in range(10000000):
                            if i%11 == 0:
                                i = i+1
                                continue # <============ example of continue
                            sum = sum + i

                        print("we get the sum: ", sum)

===============
# The else statement

        ===============
        # for + else
        ===============
        # The else part is executed when the for loop is terminated naturally.
        attendance = ["James", "Bob", "Arthur", "Helena", "Mabel"]

        for name in attendance:
            print(name)
        else:
            print("listing finished")


        # The else part is executed when the for loop is terminated naturally.
        # In other words, the loop has not been interrupted by break.
        #                 (the loop stops naturally due to no more items to loop)
        attendance = ["James", "Bob", "Arthur", "Helena", "Mabel"]

        for name in attendance:
            if name=="Helena":
                print("Helena is PRESENT")
                break
        else:
            print("Helena is ABSENT")


        attendance = ["James", "Bob", "Arthur", "Helen", "Mabel"]

        for name in attendance:
            if name=="Helena":
                print("Helena is PRESENT")
                break
        else:
            print("Helena is ABSENT")
```

```
===============
# while + else
===============
counter = 0

while counter < 3:
    counter = counter + 1
    print("Inside loop")
else:
    print("Inside else")
```

```
        ===============
        counter = 0
        while counter < 3:
            counter = counter + 1
            print("Inside loop")
            break # (1) try add break here
        else:
            print("Inside else")
        print("end of code; counter is", counter)

        ===============
        counter = 0
        while counter < 3:
            counter = counter + 1
            continue # (2) try add continue here
            print("Inside loop")
        else:
            print("Inside else")
        print("end of code; counter is", counter)
```

```
================
Break during the last iteration
================
i=0
while i<10:
    i+=1
    if i==10:
        break
else:
    print("end") # will "end" be printed?
================
attendance = ["James", "Bob", "Arthur", "Helena"]

for name in attendance:
    if name=="Helena":
        break
else:
    print("Helena is ABSENT") # will this line run?
================
```

```
==============

def check_composite(num):  # "193"
    if num.isdigit():
        num = int(num)
        for divisor in range(2, num): # divisor: 2,3,4,5...
            if not num % divisor:
                print("It is composite.")
                break
        else:
            print("It is not composite") # how to get here?
    else:
        print("Not a positive integer.") # how to get here?

# testing
check_composite("193")

for i in range(1,21):
    print(i, end=": ")
    check_composite(str(i))

        ==============
        # Change to while-loop and Run less iterations

        # 1) change "if num%divisor > 0" to "if num%divisor"

        # 2) solve for even numbers 4,6,8,... before the loop

        # 3) change to a while-loop, start from 3, stepped by 2

        # 4) instead of num-1, check until num**0.5 only
        #      Reason: For num = a * b (e.g. 15=3*5), we know that min(a,b) is at most num**0.5
        #              Think: 193**0.5 is 13.89.  To try 193 = a*b, we only consider divisors up to 13
        #
        # Ref: https://www.youtube.com/watch?v=OyvH9Drcc8s


        def check_composite(num):
            if num.isdigit():
                num = int(num)
                if num>2 and num%2==0:
                    print('It is composite.')
                else:
                    divisor = 3
                    while divisor <= num**0.5:
                        if num % divisor:
                            divisor += 2
                        else:
                            print('It is composite.')
                            break
                    else:
                        print('It is not composite.')
            else:
                print("Not valid.")

        for i in range(1, 21):
            print(i, end=": ")
            check_composite(str(i))


==============================
```