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

Python Data Types - String, List, Tuple

In this lesson, we learn 3 important data types:

- String : a sequence of characters
- List : an array of objects
- Tuple : like List, but immutable (or unchangeable)

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

Out[1]:

String (https://docs.python.org/3/library/string.html)

String is a sequence of characters. Names, Words, Sentences, Paragraphs are all examples of string.

String sequence **delimitor** is quote: (', ", """, ''')

single/double/tripple quoted string

In [2]:
```
# empty string
string_0 = ''
```

In [3]:
```
string_1 = 'This is a single-quoted string.'
print(string_1)
```

```
This is a single-quoted string.
```

In [4]:
```
string_2 = "This is a double-quoted string."
print(string_2)
```

```
This is a double-quoted string.
```

In [5]:
```
quote = "Linus Torvalds once said, \
         'Any program is only as good as it is useful.'"
print(quote)
```

```
Linus Torvalds once said,          'Any program is only as good as it is usefu
l.'
```

In [6]:
```
string_3 = '''This is a string where I
can confortably write on multiple lines
without worring about to use the escape character "\\" as in
the previsou example.
As you'll see, the original string formatting is preserved.
'''

print(string_3)
```

```
This is a string where I
can confortably write on multiple lines
without worring about to use the escape character "\" as in
the previsou example.
As you'll see, the original string formatting is preserved.
```

whitespace is blank string

The term "whitespace" refers to characters that the computer is aware of, but are invisible to human. The most common whitespace characters are spaces ("", '', ' '), tabs ('\t'), and newlines ('\n').

In [7]:
```
print("Hello python students!")
```

```
Hello python students!
```

In [8]:
```
print("Hello\tpython\t students!")
```

```
Hello   python   students!
```

In [9]:
```
print("Hello\npython\nstudents!")
```

```
Hello
python
students!
```

In [10]:
```
print("Hello\n\tpython\n\t\tstudents!")
```

```
Hello
        python
                students!
```

Strip() - removing whitespace

In [11]:
```
snake = '     python is big!   '
print(snake)
```

```
     python is big!
```

In [12]:
```
print(snake.strip())
```

```
python is big!
```

In [13]:
```
print(snake.lstrip(), ':', snake.rstrip())
```
```
python is big!    :      python is big!
```

Python cares about case

In [14]:
```
salutation = 'sir'
first_name = 'issac'
last_name = 'NEWTON'
print(salutation, first_name, last_name)
```
```
sir issac NEWTON
```

In [15]:
```
print(salutation.title(), first_name.upper(), last_name.lower())
```
```
Sir ISSAC newton
```

Note: title(), upper(), lower() are methods of string object. We will learn object/method in details later.

Concat - strings add up, and multiple too

In [16]:
```
full_name = first_name + ' ' + last_name
print(full_name)
```
```
issac NEWTON
```

In [17]:
```
greeting = 'Hello ,'
print(greeting)
```
```
Hello ,
```

In [18]:
```
# repeat a string many times
print(greeting*10)
```
```
Hello ,Hello ,Hello ,Hello ,Hello ,Hello ,Hello ,Hello ,Hello ,Hello ,
```

len() - how many characters are in a string?

In [19]:
```
len(full_name)
```
Out[19]:
```
12
```

In [20]:
```
new_line='\n'
len(new_line)
```
Out[20]:
```
1
```

String formatting

(https://docs.python.org/2/library/stdtypes.html#string-formatting)

In [21]:
```
string_template = 'The result of the calculation of {calc} is \n {res}'
print(string_template.format(calc='(3*4)+2', res=(3*4)+2))
```

```
The result of the calculation of (3*4)+2 is
 14
```

In [22]:
```
str_fmt = "%s x %s = %s" % (11, 22, 11*22)
print(str_fmt)
```

```
11 x 22 = 242
```

In [23]:
```
str_fmt1 = "{0} x {1} = {2}"

print(str_fmt1.format(10, 30, 10*30))
```

```
10 x 30 = 300
```

In [24]:
```
str_fmt2 = "{base} to the power of {exp} is equal to {pow}"

print(str_fmt2.format(exp=2, base=10, pow=10**2))
```

```
10 to the power of 2 is equal to 100
```

In [25]:
```
# print integer number
print("%d" % 100)
```

```
100
```

In [26]:
```
# print integer number in fixed length with leading 0 padding
print("%05d" % 100)
```

```
00100
```

In [27]:
```
# print float or decimal number
print("%f" % 100.135)
```

```
100.135000
```

In [28]:
```
# print float or decimal number
print("%d" % 100.135)
```

```
100
```

In [29]:
```
print("%E" % 1000000)   # print number in scientific notation
```

```
1.000000E+06
```

In [30]:
```
print("%x" % 100)   # print number in hex encoding
```

```
64
```

Why ?

$$6 \times 16^1 + 4 \times 16^0 = 100$$

unicode (https://docs.python.org/3.5/howto/unicode.html) - how to display any human lang in computer

ASCII code is for Latin western lang.

https://www.wikiwand.com/en/Unicode (https://www.wikiwand.com/en/Unicode)

```
In [31]: uni_str = "她来自中国四川，爱吃重庆火锅"
         print(uni_str)

         她来自中国四川，爱吃重庆火锅

In [32]: type(uni_str)

Out[32]: str

In [33]: byte_str = uni_str.encode()

In [34]: byte_str

Out[34]: b'\xe5\xa5\xb9\xe6\x9d\xa5\xe8\x87\xaa\xe4\xb8\xad\xe5\x9b\xbd\xe5\x9b\x9b\xe5
         \xb7\x9d\xef\xbc\x8c\xe7\x88\xb1\xe5\x90\x83\xe9\x87\x8d\xe5\xba\x86\xe7\x81\xa
         b\xe9\x94\x85'

In [35]: us1 = "我"
         bs1 = us1.encode('utf-8')
         bs1

Out[35]: b'\xe6\x88\x91'

In [36]: print(bs1.decode('utf-8'))

         我

In [37]: us2= "I"
         bs2= us2.encode('ascii')
         bs2

Out[37]: b'I'

In [38]: # ASCII code for 'I'
         ord(us2)

Out[38]: 73
```

List (https://docs.python.org/3/library/stdtypes.html#list)

List is a sequence of objects : number, character, string, object.

List sequence **delimitor** is square brackets: [,]

List may be called Array, Vector, Tensor in other lang.

In [39]:
```
empty_list = []

number_list = [-1, 0, 1]

my_shopping_list = ['Milk', 'Eggs', 'Cheese', 'Butter']
```

In [40]:
```
type(empty_list), type(number_list), type(my_shopping_list)
```

Out[40]:
```
(list, list, list)
```

string is a special list of characters

In [41]:
```
# varaible snake_name is a string
snake_name = "python"
```

In [42]:
```
type(snake_name)
```

Out[42]:
```
str
```

In [43]:
```
# varaible snake_name2 is a list
snake_name2 = ['p', 'y','t','h','o','n']
```

In [44]:
```
type(snake_name2)
```

Out[44]:
```
list
```

In [45]:
```
# convert string to list
snake_name3 = list(snake_name)
print(snake_name3)
```

```
['p', 'y', 't', 'h', 'o', 'n']
```

```
In [46]: # length of list
         len(snake_name3)

Out[46]: 6
```

index

sequence number of an item in the list

```
In [47]: # first char
         snake_name3[0]

Out[47]: 'p'
```

index is zero-based

```
In [48]: # last char
         snake_name3[-1]

Out[48]: 'n'
```

```
In [49]: snake_name3[len(snake_name3)]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-49-497d2c3c1dfa> in <module>()
----> 1 snake_name3[len(snake_name3)]

IndexError: list index out of range
```

```
In [50]: snake_name3[len(snake_name3)-1]

Out[50]: 'n'
```

```
In [51]: # going backward using negative index:
         # last char
         snake_name3[-1]

Out[51]: 'n'
```

```
In [52]: snake_name3[-3]

Out[52]: 'h'
```

- word is a list of alphabets
- sentence is a list of words and punctuations.
- paragraph is a list of sentences
- chapter ...

In [53]:
```
# python list can be made of different types
my_list = ['This', 'book', 'costs', 10.50, '$']
```

common operations / functions

len() - count list's length

In [54]:
```
len(my_list)
```
Out[54]:
```
5
```

In [55]:
```
type(my_list[-2]), type(my_list[1])
```
Out[55]:
```
(float, str)
```

index() - find an item's location

In [56]:
```
print(my_list.index('book'))
```
```
1
```

in - check existence

In [57]:
```
print('book' in my_list)
```
```
True
```

In [58]:
```
print('cost' in my_list)
```
```
False
```

append() - add more items from the back

In [59]:
```
my_list.append('I am going to order it')
```

In [60]:
```
print(my_list)
```
```
['This', 'book', 'costs', 10.5, '$', 'I am going to order it']
```

insert() - add item at a given position

In [61]:
```
my_list.insert(1, 'computer')
```

In [62]:
```
print(my_list)
```
```
['This', 'computer', 'book', 'costs', 10.5, '$', 'I am going to order it']
```

empty a list

```
In [63]: my_list = []
         print(my_list)

         []

In [64]: len(my_list)

Out[64]: 0
```

sort() - sort a list

```
In [65]: num_list = [120, 10, -1, 9999]

In [66]: num_list.sort()
         print(num_list)

         [-1, 10, 120, 9999]

In [67]: # sort reverse order
         num_list.sort(reverse=True)
         print(num_list)

         [9999, 120, 10, -1]
```

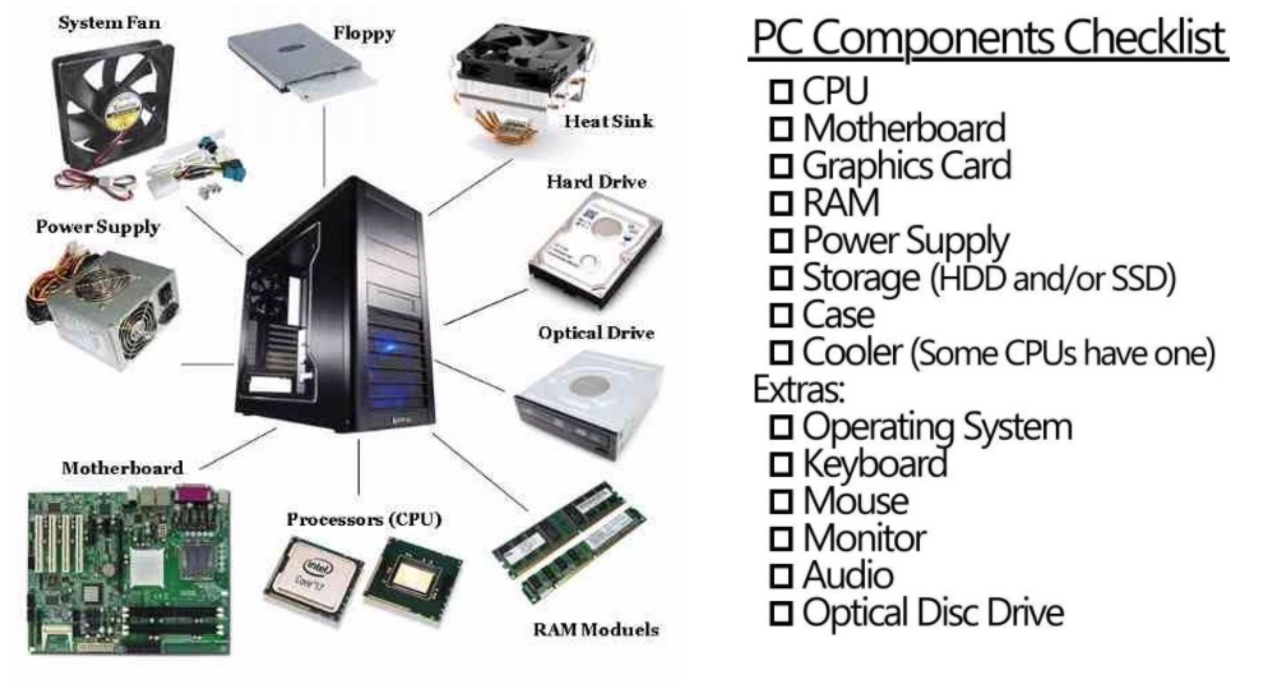
reverse() - a list

```
In [68]: num_list

Out[68]: [9999, 120, 10, -1]

In [69]: num_list.reverse()
         num_list

Out[69]: [-1, 10, 120, 9999]
```



PC Components Checklist

- ☐ CPU
- ☐ Motherboard
- ☐ Graphics Card
- ☐ RAM
- ☐ Power Supply
- ☐ Storage (HDD and/or SSD)
- ☐ Case
- ☐ Cooler (Some CPUs have one)

Extras:

- ☐ Operating System
- ☐ Keyboard
- ☐ Mouse
- ☐ Monitor
- ☐ Audio
- ☐ Optical Disc Drive

```
In [70]: PC_Components_Checklist = []
         PC_Components_Checklist.append('CPU')
         PC_Components_Checklist.append('Motherboard')
         PC_Components_Checklist

Out[70]: ['CPU', 'Motherboard']
```

extend() - a list

```
In [71]: PC_Components_Checklist.extend(['RAM', 'Power Supply', 'Hard Drive'])
```

```
In [72]: print(PC_Components_Checklist)

         ['CPU', 'Motherboard', 'RAM', 'Power Supply', 'Hard Drive']
```

```
In [73]: # what if you use append()
         PC_Components_Checklist.append(['Monitor', 'Keyboard'])
```

```
In [74]: print(PC_Components_Checklist)

         ['CPU', 'Motherboard', 'RAM', 'Power Supply', 'Hard Drive', ['Monitor', 'Keyboa
         rd']]
```

del - removing item from list

```
In [75]: del PC_Components_Checklist[-1]
```

```
In [76]: print(PC_Components_Checklist)

         ['CPU', 'Motherboard', 'RAM', 'Power Supply', 'Hard Drive']
```

pop() - remove from back

```
In [77]: last_item = PC_Components_Checklist.pop()
         print(last_item)

         Hard Drive
```

```
In [78]: print(PC_Components_Checklist)

         ['CPU', 'Motherboard', 'RAM', 'Power Supply']
```

range()

quickly generate an array - list of numbers

```
In [79]: arr = list(range(15))
         arr

Out[79]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

generate a list of words from a sentence

Strings can be split into a set of substrings when they are separated by a repeated character. If a string consists of a simple sentence, the string can be split based on spaces. The split() function returns a list of substrings. The split() function takes one argument, the character that separates the parts of the string.

In [80]:
```
split_fn_description = """
Strings can be split into a set of substrings when they are separated by a repeat
"""

word_list = split_fn_description.split()
word_list
```

Out[80]:
```
['Strings',
 'can',
 'be',
 'split',
 'into',
 'a',
 'set',
 'of',
 'substrings',
 'when',
 'they',
 'are',
 'separated',
 'by',
 'a',
 'repeated',
 'character.',
 'If',
 'a',
 'string',
 'consists',
 'of',
 'a',
 'simple',
 'sentence,',
 'the',
 'string',
 'can',
 'be',
 'split',
 'based',
 'on',
 'spaces.',
 'The',
 'split()',
 'function',
 'returns',
 'a',
 'list',
 'of',
 'substrings.',
 'The',
 'split()',
 'function',
 'takes',
 'one',
 'argument,',
 'the',
 'character',
 'that',
 'separates',
 'the',
 'parts',
 'of',
 'the',
 'string.']
```

In [81]:
```
sentence_list = split_fn_description.split('.')
sentence_list
```

Out[81]:
```
['\nStrings can be split into a set of substrings when they are separated by a
 repeated character',
 ' If a string consists of a simple sentence, the string can be split based on
 spaces',
 ' The split() function returns a list of substrings',
 ' The split() function takes one argument, the character that separates the pa
rts of the string',
 '\n']
```

Tuple (https://docs.python.org/3/library/stdtypes.html#tuple)

Tuple is a list whose item can not be changed.

Tuple sequence **delimitor** is parentheses: (,)

In [82]:
```
t = ('I', 'play', 'tennis')
```

In [83]:
```
print(t)
```

```
('I', 'play', 'tennis')
```

In [84]:
```
type(t)
```

Out[84]:
```
tuple
```

In [85]:
```
len(t)
```

Out[85]:
```
3
```

```
In [86]: t[0]

Out[86]: 'I'

In [87]: t[1]

Out[87]: 'play'

In [88]: t[2]

Out[88]: 'tennis'

In [89]: t[2] = 'ping-pong'

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-89-da0a895fd1ed> in <module>()
         ----> 1 t[2] = 'ping-pong'

         TypeError: 'tuple' object does not support item assignment


In [90]: # convert tuple to list
         lst = list(t)

In [91]: print(lst)

         ['I', 'play', 'tennis']

In [92]: type(lst)

Out[92]: list

In [93]: lst[2] = 'ping-pong'

In [94]: # convert modified list back to tuple
         tpl = tuple(lst)

In [95]: type(tpl)

Out[95]: tuple

In [96]: print(tpl)

         ('I', 'play', 'ping-pong')
```

Note: We will cover topics on looping later

```
In [ ]:

In [ ]:
```