```
# coding=utf-8
__author__ = "Nguyễn Minh Tân"
__phonenumber__ = "0987.613.161"
__copyright__ = "Copyright 2020, Nguyễn Minh Tân"

""" String Manipulation
    + Accessing Strings
    + Basic Operations
    + String Slices
    + Function and Method
"""

"""
- String hay còn gọi là xâu, hay còn gọi là chuỗi: Là một chuỗi liên tiếp các Ký tự
- Ký tự là các chữ cái, chữ số cũng như dấu, ...

Một chuỗi có thể được tạo ra bằng nhiều cách:
    - Nhập dữ liệu vào từ bàn phím với hàm input()
    - Khai báo trong cặp nháy đơn, cặp nháy kép, cặp 3 nháy đơn, cặp 3 nháy kép
    - Kết quả của các phép toán với chuỗi: cộng chuỗi, nhân chuỗi với 1 số, ...
    - Ép kiểu sang chuỗi với hàm str()
    - Đọc từ file, ...
"""
your_string = """Your string - line 1
Your string - line 2
Your string - line 3
```

```
- Multi-line - """
print(your_string)


my_string = "My string"
print(my_string)


print(my_string * 3)
print(my_string + my_string)


""" Độ dài chuỗi
    - Số lượng các ký tự trong chuỗi được gọi là chiều dài của chuỗi
    - Để tính chiều dài, chúng ta dùng hàm len()
"""
s = "123 456,789!abc"
print(len(s))  # 3 chữ cái, 9 chữ số, 1 dấy phẩy, 1 dấu chấm than và 1 dấu cách



""" Accessing Strings. Truy cập vào các ký tự trong Chuỗi.
    - Chúng ta có thể truy cập từng ký tự bằng chỉ số (index) hoặc một đoạn ký tự bằng cắt (slicing).
    - Index các ký tự trong chuỗi được đánh số từ trái qua phải và bắt đầu từ 0:
      ký tự đầu tiên là 0, ký tự thứ 2 là 1, ..., ký tự cuối cùng là (độ dài - 1)
"""
s = "Python!"
print(s[0])
print(s[1])
print(s[len(s) - 1])
```

```
print(s[len(s)])  # Lỗi vượt quá chỉ số: IndexError: string index out of range


""" Minh họa về chỉ số của chuỗi
    String s    P       y       t       h       o       n       !
    Chỉ số      0       1       2       3       4       5       6
    Truy cập    s[0]    s[1]    s[2]    s[3]    s[4]    s[5]    s[6]
"""


""" Truy cập đến ký tự trong chuỗi = chỉ số âm
    String s    P       y       t       h       o       n       !
    Chỉ số      0       1       2       3       4       5       6
    Chỉ số âm   -7      -6      -5      -4      -3      -2      -1
    Truy cập    s[0]    s[1]    s[2]    s[3]    s[4]    s[5]    s[6]
    Truy cập    s[-7]   s[-6]   s[-5]   s[-4]   s[-3]   s[-2]   s[-1]
"""
s = "Python!"
print(s[-1])
print(s[-2])
print(s[-len(s)])


print(s[-len(s) - 1])  # Lỗi vượt quá chỉ số: IndexError: string index out of range


""" Kỹ thuật cắt chuỗi - Slicing - Cơ bản
    - Dùng để lấy 1 phần từ chuỗi gốc, tạo ra chuỗi con - substring
    - Cú pháp s[i: j] trả lại 1 chuỗi con có (j-i) ký tự, bắt đầu từ ký tự có chỉ số i đến (j-1) (ko bao gồm s[j])
```

```
    => Giống kiểu làm việc của hàm range(i, j)
    - Chú ý đặc biệt:
        + s[:j]: mặc định lấy từ đầu chuỗi đến (j-1)
        + s[i:]: mặc định lấy từ i đến cuối chuỗi
        + => s[:]: trả ra chính chuỗi s
"""

s = 'Python Core'
print(s[2:8])
print(s[0:6])
print(s[7: len(s)])
print(s[-9: -1])
print(s[5: -3])

print(s[:6])
print(s[7:])
print(s[:])

"""  Kỹ thuật cắt chuỗi - Slicing - Nâng cao - Cắt với bước nhảy
    - Cú pháp s[i:j:a] trả lại chuỗi gồm các ký tự có chỉ số cách đều nhau một đoạn a, bắt đầu từ i đến trước j
    => Giống cơ chế làm việc của hàm range(i,j,a)
"""
s = 'Python Core'
print(s[::2])  # lấy các ký tự ở chỉ số chẵn
print(s[1::2])  # lấy các ký tự ở chỉ số lẻ
print(s[2:-1:3])
```

```
""" Kỹ thuật cắt chuỗi - Slicing - Nâng cao - Cắt với bước nhảy âm
    => Giống cơ chế làm việc của hàm range(i,j,a) với a là số âm
    - Cú pháp s[i:j: -a] trả lại chuỗi gồm các ký tự có chỉ số cách đều nhau một đoạn a, bắt đầu từ i lùi dần về trước j
"""
s = 'Python Core'
print(s[8:2:-2])
print(s[::-1])
print(s[-2::-3])


""" Change or Delete Chuỗi
    - Trong Python, chuỗi sau khi được tạo ra thì không thể thay đổi được (Strings are immutable).
"""
s = 'python'
s[1] = 'i'  # Lỗi: TypeError: 'str' object does not support item assignment
del s[0]  # Lỗi: TypeError: 'str' object doesn't support item deletion


""" Mã - Code của ký tự
    + Trong máy tính, mọi dữ liệu đều được lưu thành số - chính xác là dạng số nhị phân
    + Mỗi ký tự được lưu thành một số khác nhau, được gọi là mã - code
    + Danh sách các mã của các ký tự được gọi là bảng mã
    + Bảng mã đơn giản nhất là bảng mã ASCII gồm 256 ký tự, đánh số từ 0 đến 255,
    trong đó gồm tất cả các ký tự ta có thể nhìn thấy trên bàn phím máy tính
```

+ Bảng mã phổ biến nhất hiện nay là Unicode, chứa hầu hết các ngôn ngữ trên thế giới, ký tự toán học, biểu tượng cảm xúc, ...

"""

""" Bảng mã ASCII

+ Chữ cái từ A đến Z có mã tương ứng từ 65 đến 90

+ Chữ cái từ a đến z có mã tương ứng từ 97 đến 122

+ Chữ số từ 0 đến 9 có mã tương ứng từ 48 đến 57

+ Dấu cách có mã 32

"""

"""

- Lấy mã của ký tự dùng hàm ord(ký_tự_muốn_lấy_mã)

- Chuyển từ mã ra ký tự dùng hàm chr(mã_muốn_chuyển)

"""

```
print(ord('A'))
print(ord('Z'))
print(ord('b'))
print(ord('z'))
print(ord('5'))

print(chr(97))
print(chr(ord('a') - 32))
```

""" So sánh ký tự

- Do mỗi ký tự tưng ứng với mã của nó, nên ta có thể so sánh 2 ký tự với nhau.

- Kết quả của các phép so sánh chính là kết quả của so sánh 2 mã tương ứng với ký tự.

Ví dụ: 'A' < 'X' => True, 'a' < 'A' => False, 'H' == 'H' => True

"""

""" So sánh hai chuỗi với nhau bằng cách so sánh từng ký tự có trong cùng chỉ số của từng chuỗi, từ trái qua phái:

+ 1: So sánh 2 ký tự đầu tiên, ký tự của chuỗi nào lớn hơn thì chuỗi lớn hơn, nếu chúng bằng nhau thì qua bước 2

+ 2. So sánh 2 ký tự ở vị trí tiếp theo, ký tự của chuỗi nào lớn hơn thì chuỗi lớn hơn, nếu không thì chuyển qua ký tự tiếp sau.

+ 3. Cứ như vậy lần lượt so sánh các ký tự cho đến khi gặp ký tự mà 2 chuỗi khác nhau hoặc một trong hai chuỗi không còn ký tự nào.

Khi đó:

- Nếu gặp ký tự mà 2 chuỗi khác nhau, thì ký tự của chuỗi nào lớn hơn thì chuỗi lớn hơn

- Nếu cả 2 chuỗi đều không còn ký tự nào để so sánh thì 2 chuỗi đó bằng nhau

- Nếu một chuỗi không còn ký tự nào để so sánh, chuỗi còn lại vẫn còn, thì chuỗi dài hơn sẽ lớn hơn

"""

```
print('A' < 'X')
print('z' != 'Z')

s = 'programming'
print(s == 'program')
print(s > 'program')
print(s < 'pro')
print(s >= "")
```

""" Toán tử in và not in

+ in: kiểm tra xem chuỗi có nằm trong chuỗi hay không

+ not in: kiểm tra điều ngược lại với in

"""

```
s = 'Truong cao dang VienDong'
print('vien' in s)
print('Vien' in s)
print('Cao' not in s)
print('Truong cao dang' not in s)
```

""" Iterating: Duyệt các ký tự trong chuỗi

Để làm việc với từng ký tự trong chuỗi, ta có thể dùng vòng lặp for để duyệt theo chỉ số hoặc duyệt trực tiếp từng ký tự

"""

```
# Ví dụ: In ra các ký tự số trong chuỗi được nhập từ bàn phím
s = input("Nhập một chuỗi: ")
# Cách 1
for i in range(len(s)):
    if '0' <= s[i] <= '9':
        print(s[i], end="")
print()
# Cách 2:
for ky_tu in s:
    if '0' <= ky_tu <= '9':
        print(ky_tu, end="")
```

```
# Ví dụ: Lấy tất cả các vị trí của ký tự 'a' trong chuỗi
s = input("Nhập một chuỗi: ")
for i in range(len(s)):
    if s[i] == 'a':
        print(i, end=" ")



""" Các hàm phổ biến với chuỗi. Để gọi hàm trên chuỗi s ta dùng cú pháp s.tên_hàm(tham_số_nếu_cần)
    Tên hàm         Ý nghĩa
    s.upper()       Chuyển tất các chữ cái trong chuỗi s thành chữ IN HOA
    s.lower()       Chuyển tất các chữ cái trong chuỗi s thành chữ in thường
    s.title()       Viết hoa các chữ cái ở đầu các từ và viết thường các ký tự khác trong chuỗi s
    s.strip()       Xóa bỏ tất khoảng trắng ở đầu và cuối chuỗi s
    s.lstrip()      Xóa bỏ tất khoảng trắng ở đầu chuỗi s
    s.rstrip()      Xóa bỏ tất khoảng trắng ở cuối chuỗi s
    s.count(sub)        Đếm số lần xuất hiện của chuỗi sub trong chuỗi s (trả ra số nguyên)
"""

s = " Hello. i am plusPlus!    "
print(s.upper())
print(s.lower())
print(s.title())
print(s.strip())
print(s.lstrip())
print(s.rstrip())
print(s.count('us'))
```

```
print(s.count('l'))



""" Tìm kiếm và thay thế trong chuỗi
    Tên hàm             Ý nghĩa
    s.find(sub)         Trả ra chỉ số (vị trí) đầu tiên bắt đầu xuất hiện chuỗi sub bên trong chuỗi s. Nếu không có chuỗi sub trong s, trả ra -1
    s.rfind(sub)        Trả ra chỉ số cuối cùng bắt đầu xuất hiện chuỗi sub bên trong chuỗi s. Nếu không có chuỗi sub trong s, trả ra -1
    s.startswith(sub)   Trả ra True nếu chuỗi s bắt đầu bằng chuỗi sub
    s.endswith(sub)     Trả ra False nếu chuỗi s kết thúc bằng chuỗi sub
    s.replace(s1, s2)   Trả ra chuỗi mới bằng cách thay thế tất cả các chuỗi s1 trong s thành s2
"""
s = 'python is amazing'
print(s.find('on'))
print(s.find('Python'))
print(s.rfind('n'))
print(s.startswith("Py"))
print(s.endswith("ing"))
print(s.replace('is', 'are'))



""" Định dạng dữ liệu bằng f-strings
Cú pháp f-strings được giới thiệu từ Python 3.6, ngắn gọn hơn mạnh mẽ hơn, cho phép chèn giá trị biến hoặc nhúng cả biểu thức vào trong chuỗi
"""
```

```
import math
name, age = 'Python', '30'
a, b = 3, 4.5

print(f"{name} is {age} years old.")
print(f"PI number: {math.pi:10.4}")  # ko nói gì thì mặc định là 6 chữ số thập phân, nhưng ở đây lấy 4
print(f"Diện tích hình chữ nhật {a}x{b} = {a*b:.3}")

from datetime import datetime
today = datetime.today()
print(f"Today: {today:%B %d, %Y}")


""" Bài tập về nhà:
Bài 01: Viết chương trình thay thế tất cả các ký tự giống ký tự đầu tiên của một Chuỗi thành $.
Bài 02: Viết chương trình để xóa bỏ ký tự thứ m trong chuỗi không rỗng, với m là số không âm, nhập từ bàn phím.
Bài 03: Viết chương trình để xóa bỏ các ký tự có chỉ số là số lẻ trong một chuỗi.
Bài 04: Viết chương trình sinh ra một chuỗi từ 2 ký tự đầu và 2 ký tự cuối trong chuỗi được nhập vào,
    nếu độ dài chuỗi nhỏ hơn 2 thì in ra chuỗi rỗng.
Bài 05: Viết chương trình tìm ra ký tự lớn nhất và ký tự nhỏ nhất của một chuỗi nhập từ bán phím.
Bài 06: Viết chương trình đảo ngược từ ký tự thường sang ký tự hoa và từ ký tự hoa sang ký tự thường trong một chuỗi.
"""
```