1. partition(str): Hàm trả về 1 đối tượng tuple, trong đó chuỗi được chia thành 3 phần

+ Part 1: tất cả mọi thứ ở trước chuỗi str
+ Part 2: chuỗi str
+ Part 3: phần ở sau chuỗi str

In [ ]:
```
a = "I have a banana and I like eat banana."
b = a.partition("have")
print(b)
```

```
('I ', 'have', ' a banana and I like eat banana.')
```

2. replace(oldValue, newValue): Hàm trả về chuỗi, trong đó giá trị oldValue trong chuỗi được thay thế bằng giá trị newValue.

In [ ]:
```
b = a.replace("a","w")
print(b)
```

```
I hwve w bwnwnw wnd I like ewt bwnwnw.
```

3. rfind(searchValue, start, end): Hàm trả về vị trí của lần xuất hiện cuối cùng của giá trị searchValue trong chuỗi từ vị trí start đến vị trí end. Nếu không tìm thấy giá trị searchValue thì trả về -1. (Hàm này tương tự hàm rindex())

- searchValue: giá trị cần tìm
- start: vị trí bắt đầu tìm, mặc định là 0
- end: vị trí kết thúc tìm, mặc định là vị trí kết thúc của chuỗi

In [ ]:
```
b = a.rfind('banana')
print(b)
```

```
31
```

4. rjust(lenOfString, charToJustify): Hàm trả về chuỗi được căn lề phải

+ lenOfString: tổng độ dài của chuỗi
+ charToJustify: kí tự dùng để justify chuỗi

In [ ]:
```
a = "Hello"
b = a.rjust(10, '-')
print(b)
```

```
-----Hello
```

5. rpartition(searchValue): Hàm tìm kiếm lần xuất hiện cuối cùng của giá trị searchValue trong chuỗi, và cắt chuỗi thành 3 phần tương tự như hàm partition()

+ Part 1: tất cả mọi thứ ở trước searchValue
+ Part 2: searchValue
+ Part 3: phần ở sau searchValue

In [ ]:
```
a = "Becuase I don't know programming, so I word hard than others."
b = a.rpartition("so")
print(b)
```

```
("Becuase I don't know programming, ", 'so', ' I word hard than others.')
```

6. rsplit(separator, maxSplit): Hàm sẽ tách chuỗi dựa trên ký tự separator, và trả về một list.

+ separator: Ký tự dùng để phân tách chuỗi
+ maxSplit: Số lượng split tối đa có thể thực hiện. Nếu maxSplit = -1 thì hàm sẽ split toàn bộ

In [ ]:
```
a = "apple, banane, orange, lemon"
b = a.rsplit(", ")
print(b)
```

```
['apple', 'banane', 'orange', 'lemon']
```

In [ ]:
```
b = a.rsplit(", ", 2)
print(b)
```

```
['apple, banane', 'orange', 'lemon']
```

7. rstrip(splitStr): Hàm trả về chuỗi được xóa splitStr dư thừa (splitStr phải ở cuối chuỗi, mặc định splitStr là khoảng trắng).

In [ ]:
```
a = "      banana     "
b = a.rstrip()
print("of all fruits",b, "is my favorite")
```

```
of all fruits       banana is my favorite
```

8. split(separator): Hàm sẽ tách chuỗi dựa trên separator, và trả về list.

In [ ]:
```
a = "apple, banana, orange"
b = a.split(", ")
print(b)
```

```
['apple', 'banana', 'orange']
```

In [ ]:
```
b = a.split()
print(b)
```

```
['apple,', 'banana,', 'orange']
```

9. splitlines(keepLineBreak): Hàm sẽ tách chuỗi tại vị trí ngắt dòng (line break) và trả về list

+ keepLineBreak: Giữ lại ký tự \n nếu có giá trị là True, mặc định là False

In [ ]:
```
a = """Thank you for the music
Welcome to the jungle"""
b = a.splitlines()
print(b)
```

```
['Thank you for the music', 'Welcome to the jungle']
```

In [ ]:
```
b = a.splitlines(True)
print(b)
```

```
['Thank you for the music\n', 'Welcome to the jungle']
```

10. startwith(startStr, start, end): Hàm trả về True nếu chuỗi bắt đầu bằng startStr từ start đến end, ngược lại thì trả về False

+ startStr: giá trị để kiểm tra xem chuỗi có bắt đầu bằng nó không
+ start: vị trí bắt đầu tìm kiếm startStr
+ end: vị trí kết thúc tìm kiếm startStr

In [ ]:
```
a = "Hello, welcome to my world."
b = a.startswith("Hello")
print(b)
```

```
True
```

In [ ]:
```
b = a.startswith("welcome")
print(b)
```

```
False
```

In [ ]:
```
b = a.startswith("welcome", 7)
print(b)
```

```
True
```

In [ ]:
```
b = a.startswith("welcome", 7, 20)
print(b)
```

```
True
```

11. strip(chars): Hàm sẽ xóa chars dư thừa ở đầu và cuối chuỗi và trả về chuỗi sau khi xóa

In [ ]:
```
a = ",,,,,rrttgg.....banana....rrr"
b = a.strip(",.grt")
print(b)
```

```
banana
```

12. swapcase(): Hàm chuyển đổi từ chuỗi in thường sang chuỗi in hoa, và ngược lại

In [ ]:
```
a = "hello"
b = a.swapcase()
print(b)
```

```
HELLO
```

In [ ]:
```
a = "HELLO"
b = a.swapcase()
print(b)
```

```
hello
```

13. title(): Hàm sẽ biến đổi ký tự đầu tiên của mỗi từ trong chuỗi sang ký tự in hoa

In [ ]:
```
a = "Welcome to my world"
b = a.title()
print(b)
```

```
Welcome To My World
```

14. str.translate(table): Hàm sẽ trả về chuỗi đã được dịch.

- table: Có thể là đối tượng dictionary hoặc map, nó sẽ mô tả cách thực hiện thay thế các ký tự trong chuỗi bằng các ký tự tương ứng trong table (table được tạo ra khi sử dụng hàm maketrans())

In [ ]:
```
a = "Hello Sam!"
table = str.maketrans("S", "P")
print(a.translate(table))
```

```
Hello Pam!
```

In [ ]:
```
table = str.maketrans("Ham", "Gys")
print(a.translate(table))
```

```
Gello Sys!
```

15. upper(): Hàm trả về chuỗi với các ký tự dược in hoa.

In [ ]:
```
a = "heLLo"
b = a.upper()
print(b)
```

```
HELLO
```

16. zfill(lenOfStr): Hàm sẽ điền vào chuỗi với số lượng số 0 ở vị trí bắt đầu chuỗi

+ lenOfStr: tổng độ dài chuỗi

In [ ]:
```
a = "55"
b = a.zfill(10)
print(b)
```

```
0000000055
```

In [ ]:
```
a = "hello"
b = "welcome to the jungle"
c = "10.000"

print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))
```

```
00000hello
welcome to the jungle
000010.000
```