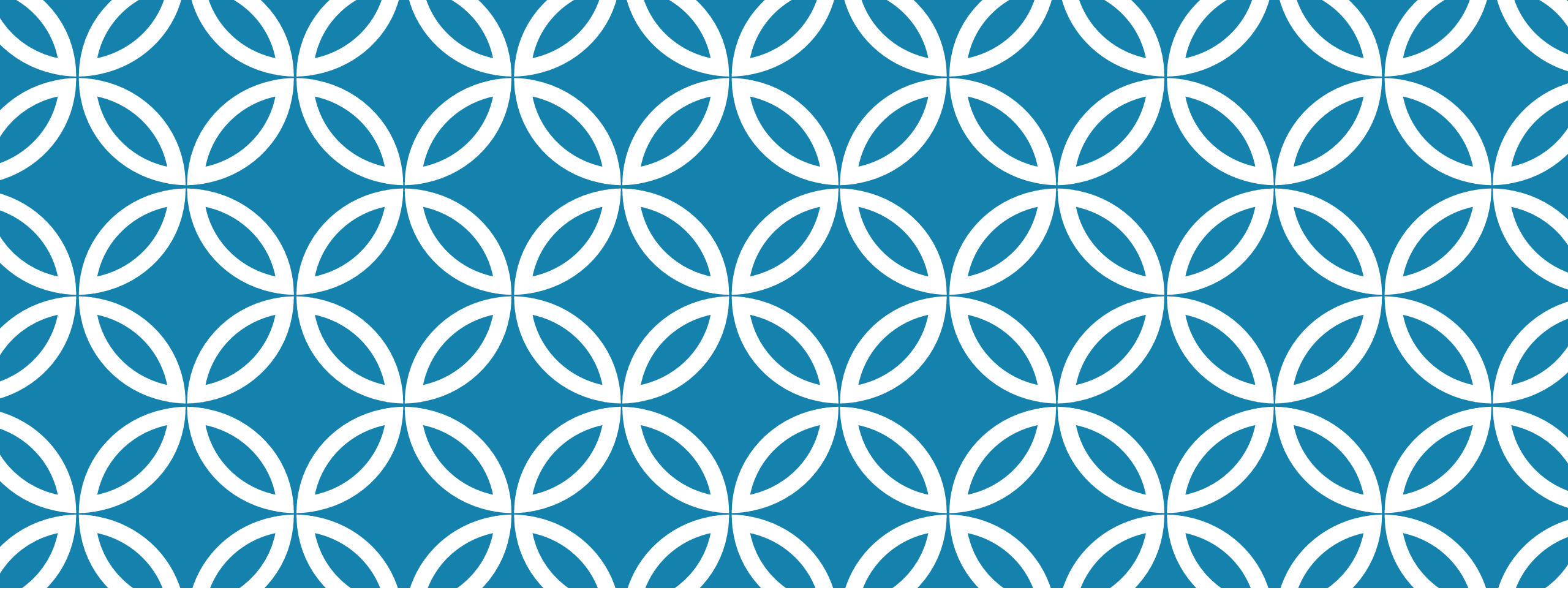


PYTHON STRING

01418112 Fundamental Programming Concepts

STRING DATA TYPE

งานหลักสำคัญอย่างหนึ่งของคอมพิวเตอร์ คือ การประมวลผลข้อความ **(text processing)**

Search engine อย่างเช่น **Google** เป็นตัวอย่างสำคัญของโปรแกรมประยุกต์ที่ทำหน้าที่ประมวลผลข้อความ

string เป็นลำดับของอักขระ **(sequence of characters)**

string เป็นชนิดข้อมูลในภาษาโปรแกรมในปัจจุบัน ที่ทำหน้าที่จัดเก็บและประมวลผลข้อมูลที่อยู่ในรูปของข้อความ

ค่าคงที่ **string (string literals)** ได้แก่ ลำดับของอักขระที่เขียนอยู่ระหว่างอักขระ **single quote (')** หรืออักขระ **double quotes (")** ที่จับคู่ตรงกัน เพื่อเปิดและปิดค่าคงที่ **string**

STRING DATA TYPE

```
>>> str1 = 'Good Morning'
>>> str2 = "Goodbye"
```

ตัวแปร **str1** และ **str2** เป็นตัวแปรที่อ้างอิง **string objects** ในที่นี้คือค่าคงที่ **string 'Good Morning'** และ **"Goodbye"** ตามลำดับ

```
>>> print(str1, str2)
>>> Good Morning Goodbye
>>> type(str1)
>>> <class 'str'>
```

STRING DATA TYPE

อักขระ **single quotes** จะต้องถูกกำกับด้วยอักขระ **backslash (\)** ภายใน **single quoted string:**

```
>>> str3 = 'I don\'t like rat.'
```

ทำนองเดียวกัน ต้องกำกับอักขระ **double quote** ด้วยอักขระ \ ภายใน **double quoted string:**

```
>>> str4 = "She said: \"I will come back tomorrow.\""
```

ยกเว้น

```
>>> str5 = "It can\'t jump."
```

อักขระ \ ใช้กำกับอักขระที่ถูกนิยามให้มีความหมายพิเศษ เมื่อต้องการใช้อักขระดังกล่าวในความหมายปกติ

STRING DATA TYPE

```
>>> txt = '''A string in triple quotes can extend
over multiple lines like this one, and can contain
'single' and "double" quotes.'''
```

```
>>> print(txt)
A string in triple quotes can extend
over multiple lines like this one, and can contain
'single' and "double" quotes.
```

ระหว่าง **triple-quoted strings** เปิดและปิด สามารถใช้อักขระ **newlines** และ **quotes** โดยไม่ต้องกำกับด้วย \

SOME OPERATIONS & FUNCTIONS

Concatenation

สามารถเชื่อม **2 strings** เข้าด้วยกันตามลำดับจากซ้ายไปขวา โดยใช้ตัวดำเนินการ **+**
นิพจน์ **"Hello" + "World"** มีค่าเท่ากับ **"HelloWorld"**

Repetition

สามารถเชื่อม **string** เดิมซ้ำ ๆ กัน ในลักษณะเดียวกับการเชื่อม **string** เดิมเข้าด้วยกัน ทำโดยใช้ตัวดำเนินการ *
นิพจน์ **"*-*" * 3** มีค่าเท่ากับ **"*-**-**-*"**

Indexing

สามารถเข้าถึงอักขระต่าง ๆ ใน **string** โดยการระบุดัชนี **(index)**
นิพจน์ **"Python"[0]** มีค่าเท่ากับ **"P"**

SOME OPERATIONS & FUNCTIONS

Slicing

สามารถสร้าง string ย่อย (substrings) โดยการ slice ในลักษณะเดียวกับ List data type
นิพจน์ "Python"[2:4] มีค่าเท่ากับ "th"

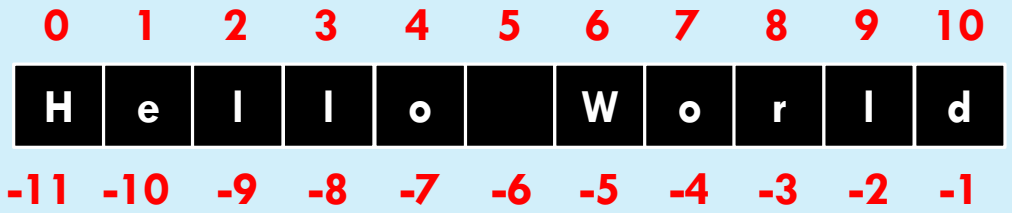
Size

สามารถหาขนาดหรือความยาวของ string ด้วยฟังก์ชัน len
นิพจน์ len("Python") มีค่าเท่ากับ 6

SOME OPERATIONS & FUNCTIONS

อักขระลำดับแรกใน **string** มีดัชนี (**index**) เป็น **0** เช่นเดียวกับ **List data type**

```
>>> s = "Hello World"
>>> s[0]
'H'
>>> s[5]
' '
>>> s[len(s)-1]
'd'
>>> s[-2]
'l'
```



IMMUTABLE STRINGS

```
>>> s = "Strings are immutable!"
>>> s[-1] = '.'
Traceback (most recent call last):
  Python Shell, prompt 17, line 1
builtins.TypeError: 'str' object does not support item assignment
```

จากตัวอย่างจะเห็นได้ว่า **strings** ไม่สามารถเปลี่ยนแปลงค่าได้ **(immutable)**

```
>>> txt = "Today is Monday."
>>> txt = "Tomorrow is Tuesday."
```

คำสั่งกำหนดค่าลำดับที่สองเปลี่ยนให้ตัวแปร **txt** อ้างอิงไปที่ค่าคงที่ **string** ใหม่

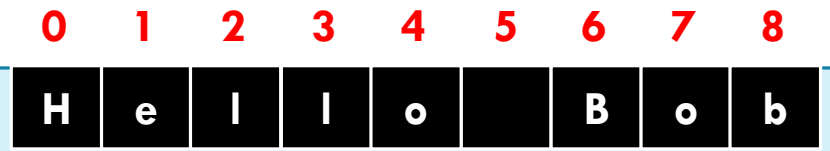
–> ไม่ได้หมายความว่าตัวแปร **txt** เปลี่ยนแปลงค่า

IMMUTABLE STRINGS

```
>>> a = "Program"
>>> b = "Program"
>>> c = b
>>> a is b
True
>>> a is c
True
>>> b is c
True
```

ในที่นี้ตัวแปร **a, b, c** อ้างอิงไปที่ค่าคงที่ **string "Program"** เดียวกัน

MORE STRING SLICING




```
>>> greet = 'Hello Bob'
>>> greet[0:3]
'Hel'
>>> greet[5:9]
' Bob'
>>> greet[:5]
'Hello'
>>> greet[5:]
' Bob'
>>> greet[:]
'Hello Bob'
>>> greet[::2]
' HloBb'
```

IN OPERATORS

ใช้ตัวดำเนินการ in ร่วมกับ for เพื่อให้เกิดการวนเข้าถึงแต่ละอักขระในแต่ละรอบของการวนซ้ำ

```
>>> for ch in "Spam!":
        print (ch, end=" ")
S p a m !
```

ใช้ตัวดำเนินการ in เพื่อตรวจสอบการเป็น (หรือไม่เป็น) substring ของอีก string หนึ่ง

```
>>> if '1' not in "02468":
        print("Yes")
Yes
```

STRING PROCESSING EXAMPLE I

```
# username.py
# get user's first and last names
first = input("Please enter your first name (all lowercase): ")
last = input("Please enter your last name (all lowercase): ")

# concatenate first initial with 7 chars of last name
uname = first[0] + last[:7]
print(uname)
```

STRING PROCESSING EXAMPLE I

ผลการรันโปรแกรมครั้งที่ 1

```
Please enter your first name (all lowercase): gui van
Please enter your last name (all lowercase): rossum
grossum
```

ผลการรันโปรแกรมครั้งที่ 2

```
Please enter your first name (all lowercase): donald
Please enter your last name (all lowercase): trump
dtrump
```

STRINGS AND LISTS

Strings เป็นลำดับของอักขระ ขณะที่ **lists** เป็นลำดับของค่าที่มีชนิดได้หลากหลาย
Lists เป็นลำดับของตัวเลข **strings** หรือทั้งสองอย่าง เช่น **myList = [1, "Spam ", 4, "U"]**
Lists สามารถเปลี่ยนแปลงค่าได้ **(mutable)** ขณะที่ **strings** ไม่สามารถเปลี่ยนแปลงค่าได้

```
>>> ls1 = [2, 14, 38, 45, 16]
>>> ls1[3] = 0
>>> ls1
[2, 14, 38, 0, 16]
>>> lst1[:4] = [5, 18, 40]
>>> ls1
[5, 18, 40, 16]
```

MORE STRING FUNCTIONS

ฟังก์ชัน **ord(ch)** คืนค่าจำนวนเต็มที่เป็นลำดับของอักขระ **ch** ใน **ASCII table**

ฟังก์ชัน **chr(x)** คืนค่าอักขระที่ตรงกับจำนวนเต็ม **x** ใน **ASCII table**

```
>>> ord("A")
65
>>> ord("a")
97
>>> chr(97)
'a'
>>> chr(65)
'A'
```

STRING PROCESSING - EXAMPLE II - ENCODING

ใช้ฟังก์ชัน **ord()** และ **chr()** ในการเข้ารหัส **(encode)** และถอดรหัส **(decode)** ข้อความที่นำเข้า

ขั้นตอนวิธีในเข้ารหัส **(encoder algorithm)** มีดังนี้:

1. get the message to encode
2. for each character in the message:
    2.1 print the letter number of the character

STRING PROCESSING - EXAMPLE II - ENCODING

```
# A program to convert a textual message into a sequence of numbers,
# utilizing the underlying Unicode encoding.


print("This program converts a textual message into a sequence")
print("of numbers representing the Unicode encoding of the message.\n")
message = input("Please enter the message to encode: ")
print("\nHere are the Unicode codes:")
for ch in message:
    print(ord(ch), end=" ")
print()
```

STRING PROCESSING - EXAMPLE II - DECODING

ขั้นตอนวิธีในถอดรหัส (decoder algorithm) มีดังนี้:

```
1. get the sequence of numbers to decode
2. message = ""          # empty string
3. for each number in the input:
       3.1 convert the number to the appropriate character
       3.2 add the character to the end of the message
4. print the message
```

เมื่อเริ่มต้นตัวแปร **message** ถูกกำหนดให้มีค่าเป็น **string** ว่าง จากนั้นในแต่ละรอบของ **for loop** ตัวแปร **message** จะทำหน้าที่สะสม **character** ที่ได้จากการถอดรหัสเลขหนึ่งจำนวน โดยเพิ่ม **character** ดังกล่าวต่อจากอักขระตัวสุดท้ายของ **message**

STRING PROCESSING - EXAMPLE II - DECODING

คำถาม: ขั้นตอน get the sequence of number to decode นั้นจะทำได้อย่างไร

คำตอบ: อ่านลำดับของตัวเลขที่ได้จากการเข้ารหัสในรูปของ **string** แล้วแบ่ง **string** ดังกล่าวออกเป็น **string** ย่อย แต่ละส่วนย่อยเป็นเหนึ่งจำนวนที่จะนำมาถอดรหัส

ปรับปรุงขั้นตอนวิธีในถอดรหัสเป็นดังนี้:

```
1. get the sequence of numbers as a string, inString
2. message = ""          # empty string
3. for each of the smaller strings:
       3.1 change the string of digits into the number it represents
       3.2 append the ASCII character for that number to message
4. print message
```

STRING METHOD

ข้อมูลชนิด **string** มี **method** ที่ทำหน้าที่แบ่ง **string** ออกเป็นชิ้นย่อย ๆ คือ **method split**

ตัวอย่างต่อไปนี้จะ **split string** เป็นส่วนย่อยตามช่องว่างภายใน **string**

```
>>> "Hello string methods!".split()
['Hello', 'string', 'methods!']
```

นอกจากนี้ เราสามารถ **split string** ออกเป็นส่วนย่อยตาม **character** ที่กำหนดให้ได้ดังนี้

```
>>> "32,24,25,57".split(",")
['32', '24', '25', '57']
```

STRING PROCESSING - EXAMPLE II - DECODING

```
# A program to convert a sequence of Unicode numbers into a string of text
print("This program converts a sequence of Unicode numbers into")
print("the string of text that it represents.\n")

inString = input("Please enter the Unicode-encoded message: ")
message = ""
for numStr in inString.split():
    # convert the (sub)string to a number
    codeNum = int(numStr)
   # append character to message
    message = message + chr(codeNum)

print("\nThe decoded message is:", message)
```

STRING PROCESSING - EXAMPLE II

ผลลัพธ์จากการรับโปรแกรมในส่วน **encoder** และ **decoder** ต่อเนื่องกัน:

```
----------------------------------------------------------------------
This program converts a textual message into a sequence
of numbers representing the Unicode encoding of the message.
Please enter the message to encode: Hello
Here are the Unicode codes: 72 101 108 108 111

----------------------------------------------------------------------
This program converts a sequence of Unicode numbers into
the string of text that it represents.
Please enter the ASCII-encoded message: 72 101 108 108 111
The decoded message is: Hello
```

STRING PROCESSING – INPUT/OUTPUT

สมมติถ้าต้องการจัดรูปแบบวันที่จาก **"05/24/2003"** ให้แสดงผลในรูปแบบ **"May 24, 2003."** จะสามารถทำได้อย่างไร

พิจารณาขั้นตอนในการแก้ปัญหาดังต่อไปนี้

1. Input the date in mm/dd/yyyy format (dateStr)
2. Split dateStr into month, day, and year strings
3. Convert the month string into a month number
4. Use the month number to lookup the month name
5. Create a new date string in the form “Month Day, Year”
6. Output the new date string

STRING PROCESSING – INPUT/OUTPUT

```
dateStr = input("Enter a date (mm/dd/yyyy): ")
monthStr, dayStr, yearStr = dateStr.split("/")
```

ข้อมูลวันที่ถูกนำเข้าในรูปของข้อมูล **string** จากนั้นใช้ **method split()** เพื่อแยกออกเป็น **string** ย่อย เดือน วัน และปี ตาม **string "/"** พร้อมกับกำหนดเป็นค่าของตัวแปร **3** ตัว ประกอบด้วย **monthStr, dayStr** และ **yearStr** ตามลำดับ

MORE STRING METHODS (1)

s.capitalize()

คืนค่าสตริงสำเนาของ s ที่อักขระแรกเป็นตัวพิมพ์ใหญ่

s.title()

คืนค่าสตริงสำเนาของ s ที่อักขระแรกของแต่ละคำเป็นตัวพิมพ์ใหญ่

```
s = "this is string example....wow!!!"
print("s.capitalize() :", s.capitalize())
s.capitalize() : This is string example....wow!!!
print("s.title()      :", s.title())
s.title()      : This Is String Example....Wow!!!
```

```
12345678901234567890123456789012345678901234567890
```

MORE STRING METHODS (2)

s.center(width[, fillchar])

คืนค่าสตริงใหม่ที่ s ถูกจัดวางในตำแหน่งกึ่งกลางระหว่างพื้นที่ขนาด width โดยเติมเต็มพื้นที่ส่วนที่เหลือมีค่าเป็นช่องว่างหรือ fillchar (ถ้าระบุ)

```
s = "this is string example..."
print("s.center(30)      :", s.center(30))
s.center(30)      :   this is string example...
print("s.center(30, '*') :", s.center(30, '*'))
s.center(30, '*') : **this is string example...***
```

```
123456789012345678901234567890123456789012345678901234567890
```

MORE STRING METHODS (3)

s.count(sub, start= 0, end=len(string))

คืนค่าจำนวนครั้งที่สตริงย่อย **sub** ปรากฏใน (**slicing** [**start**, **end**] ของ) สตริง **s**

```
s = "this is string example....wow!!!"
sub = "i"
print("s.count(sub, 4, 40) :", s.count(sub, 4, 40))
s.count(sub, 4, 40) : 2
sub = "wow"
print("s.count(sub) :", s.count(sub))
s.count(sub) : 1
```

```
1234567890123456789012345678901234567890
```

MORE STRING METHODS (4)

s.startswith(prefix, beg=0, end=len(s))

คืนค่า True เมื่อสตริง s[beg:end] ขึ้นต้นด้วยสตริง prefix มิฉะนั้นคืนค่า False

```
s = "this is string example....wow!!!"
print(s.startswith('this'))
True
print(s.startswith('is', 2, 4))
True
print(s.startswith('this', 2, 4))
False
```

MORE STRING METHODS (5)

s.endswith(suffix[, start[, end]])

คืนค่า **True** เมื่อสตริง **s** สิ้นสุดด้วยสตริง **suffix** มิฉะนั้นคืนค่า **False** หรือสิ้นสุดใน **slicing** [**start**, **end**] ของสตริง **s**

```
s = "this is string example....wow!!!"
suffix = "wow!!!"
print(s.endswith(suffix), s.endswith(suffix,20))
True True
print(s.endswith("is", 2, 4), s.endswith("is", 2, 6))
False False
```

MORE STRING METHODS (6)

s.find(str1, beg=0, end=len(s))

คืนค่าตำแหน่งที่สตริง str1 ปรากฏเป็นครั้งแรกในสตริง s[beg:end] หรือคืนค่า -1 ในกรณีที่ไม่พบสตริง str1 ในสตริง s

```
s = "this is string example....wow!!!"
s2 = "is"
print(s1.find(s2))
2
print(s1.find(s2, 3))
5
print(s1.find(s2, 20))
-1
```

MORE STRING METHODS (7)

s.rfind(str1, beg=0, end=len(s))

คืนค่าตำแหน่งที่สตริง str1 ปรากฏเป็นครั้งแรกในสตริง s[beg:end] เมื่อเริ่มหาจากด้านท้ายสตริง หรือคืนค่า -1 ในกรณีที่ไม่พบสตริง str1 ในสตริง s

```
s = "this is string example....wow!!!"
print(s.rfind('is'), s.rfind('is', 0, 10))
5 5
print(s.rfind('is', 10, 0), s.find('is', 10, 0))
-1 -1
print(s.find('is'), s.find('is', 0, 10))
2 2
```

MORE STRING METHODS (8)

s.replace(old, new[, max])

คืนค่าสตริงใหม่ที่ทุกตำแหน่งของสตริงย่อย **old** ถูกแทนที่ด้วยสตริงย่อย **new** และจำกัดจำนวนครั้งของการแทนที่เท่ากับ **max**

```
s = "this is string example, this is real string"
print(s.replace("is", "was"))
thwas was string example, thwas was real string
print(s.replace("is", "was", 3))
thwas was string example, thwas is real string
```

```
1234567890123456789012345678901234567890
```

MORE STRING METHODS (9)

s.join(sequence)

คืนค่าสตริงใหม่ที่ได้จากการเชื่อมสตริงที่อยู่ใน **sequence** โดยคั่นด้วยสตริง **s**

```
s = "-"
seq = ("a", "b", "c") # This is sequence of strings
print(s.join(seq))
a-b-c
```

MORE STRING METHODS (10)

s.lstrip([char])

คืนค่าสตริงใหม่ที่ได้จากการลบทุก **char** ที่ปรากฏอยู่ที่ส่วนต้นของสตริง **s** ในกรณีที่ไม่ระบุ **char** อักขระ **space** จะถูกลบ

s.rstrip([char])

คืนค่าสตริงใหม่ที่ได้จากการลบทุก **char** ที่ปรากฏอยู่ที่ส่วนท้ายของสตริง **s** ในกรณีที่ไม่ระบุ **char** อักขระ **space** จะถูกลบ

MORE STRING METHODS (11)

```
s = "      this is string example....wow!!!      "
print(s.lstrip())
this is string example....wow!!!
print(s.rstrip())
this is string example....wow!!!
s = "88888888this is string example....wow!!!8888888"
print(s.lstrip('8'))
this is string example....wow!!!8888888
print(s.rstrip('8'))
88888888this is string example....wow!!!
```

```
1234567890123456789012345678901234567890
```

MORE STRING METHODS (12)

s.ljust(width[, fillchar])

คืนค่าสตริงใหม่ที่ได้จัดวาง s ในตำแหน่งชิดด้านซ้ายของพื้นที่ขนาด width โดยเติมเต็มพื้นที่ส่วนที่เหลือมีค่าเป็นช่องว่างหรือ fillchar (ถ้าระบุ)

s.rjust(width[, fillchar])

คืนค่าสตริงใหม่ที่ได้จัดวาง s ในตำแหน่งชิดด้านขวาของพื้นที่ขนาด width โดยเติมเต็มพื้นที่ส่วนที่เหลือมีค่าเป็นช่องว่างหรือ fillchar (ถ้าระบุ)

```
s = "this is string example....wow!!!"
print(s.ljust(50, '0'))
this is string example....wow!!!00000000000000000
print(s.rjust(50, '0'))
00000000000000000this is string example....wow!!!
```

MORE STRING METHODS (13)

s.isalnum()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็น **alphanumeric** และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้นคืนค่า **False**

s.isalpha()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็น **alphabetic** และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้น คืนค่า **False**

s.isdigit()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็น **digit** และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้นคืนค่า **False**

MORE STRING METHODS (14)

:: **Alphanumeric character** ประกอบด้วยอักขระ **62** ตัว คือ **A-Z, a-z** และ **0-9**

:: **Alphabetic character** ประกอบด้วยอักขระ **52** ตัว คือ **A-Z** และ **a-z**

:: **Digit character** ประกอบด้วยอักขระ **10** ตัว คือ **0-9**

```
s = "this2017"
print(s.isalnum(), s.isalpha(), s.isdigit())
True False False

s = "This"
print(s.isalnum(), s.isalpha(), s.isdigit())
True True False
```

MORE STRING METHODS (15)

```
s = "2017"
print(s.isalnum(), s.isalpha(), s.isdigit())
True False True

s = "20.15"
print(s.isalnum(), s.isalpha(), s.isdigit())
False False False

s = "this is string example....wow!!!"
print(s.isalnum(), s.isalpha(), s.isdigit())
False False False
```

MORE STRING METHODS (16)

s.isspace()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็นอักขระ **whitespace** และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้นคืนค่า **False**

```
s = "     "
print(s.isspace())
True

s = "this is string example....wow!!!"
print(s.isspace())
False
```

MORE STRING METHODS (17)

s.islower()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็นอักขระตัวพิมพ์เล็ก และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้นคืนค่า **False**

s.isupper()

คืนค่า **True** ถ้าทุกอักขระใน **s** เป็นอักขระตัวพิมพ์ใหญ่ และ **s** มีความยาวอย่างน้อย **1** อักขระ มิฉะนั้นคืนค่า **False**

```
s = "THIS is string example....wow!!!"
print(s.islower(), s.isupper())
False False
```

MORE STRING METHODS (18)

```
s = "This is string example....wow! 555"
print(s.islower(), s.isupper())
False False

s = "this is string example"
print(s.islower(), s.isupper())
True False

s = "THIS IS STRING EXAMPLE"
print(s.islower(), s.isupper())
False True
```

MORE STRING METHODS (19)

s.lower()

คืนค่าสตริงใหม่ที่ได้จากการแปลงอักขระที่เป็นตัวพิมพ์ใหญ่ใน s ให้เป็นอักขระตัวพิมพ์เล็กทุกอักขระ

s.upper()

คืนค่าสตริงใหม่ที่ได้จากการแปลงอักขระที่เป็นตัวพิมพ์เล็กใน s ให้เป็นอักขระตัวพิมพ์ใหญ่ทุกอักขระ

```
s = "THIS IS STRING EXAMPLE....wow!!! 555"
print(s.lower())
this is string example....wow!!! 555
print(s.upper())
THIS IS STRING EXAMPLE....WOW!!! 555
```

MORE STRING METHODS (20)

s.swapcase()

คืนค่าสตริงใหม่ที่ได้จากการแปลงอักขระที่เป็นตัวพิมพ์ใหญ่ใน **s** ให้เป็นอักขระตัวพิมพ์เล็กทุกอักขระ และแปลงอักขระที่เป็นตัวพิมพ์เล็กใน **s** ให้เป็นอักขระตัวพิมพ์ใหญ่ทุกอักขระ

```
s = "THIS IS STRING EXAMPLE....wow!!! 555"
print(s.swapcase())
this is string example....WOW!!! 555
```

MORE STRING METHODS (21)

s.zfill(width)

คืนค่าสตริงใหม่ที่ได้จากการเติมอักระ 0 ทางด้านซ้ายของสตริง s เพื่อให้เต็มความยาว width

```
s = "this is string example....wow!!!" # len(s) -> 32
print(s.zfill(35))
000this is string example....wow!!!
print(s.zfill(40))
00000000this is string example....wow!!!
print(s.zfill(30))
this is string example....wow!!!
```

```
12345678901234567890123456789012345678901234567890
```

REFERENCES

1. ดัดแปลงจากเอกสาร **Python Programming: An Introduction to Computer Science**
2. **http://www.python-course.eu/python3_variables.php**
3. **http://www.tutorialspoint.com/python/python_strings.htm**