



PYTHON(Day3)

Natthaporn Phongvijit (SDI Team)



CONTENT

String

Datetime

Try Except



“String”

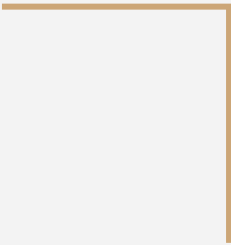


Content of String

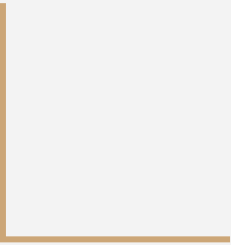
- Index of string
- String Methods

What is String?

คือ ชนิดของข้อมูลที่มีลักษณะเป็นตัวอักษรต่อกันยาวๆ หรือ เป็น
รูปประโยค



“Index of string”



Index of string

- คือ การเข้าถึงตำแหน่งต่างๆของ String ผ่านทาง Index โดยปกติแล้วตำแหน่งต่างๆของ String จะเริ่มต้นจาก ตำแหน่งที่ 0 ไล่ไปจนถึงตำแหน่งสุดท้าย

Index	0	1	2	3	4	5
String	P	y	t	h	o	n

Index of string

➤ คือ การเข้าถึงตำแหน่งต่างๆของ String ผ่านทาง Index

Ex:

```
data = "Python"
```

```
print data[0] #P
```

```
print data[2] #t
```


Index of string

- การตัดคำด้วย Index คือการเอาเฉพาะส่วนของคำ แสดงดังนี้

Ex:

```
data = "python"
```

```
print data[0:2] #py
```

```
print data[1:4] #yth
```

Length of string

➤ การหาความยาวหรือขนาดของ String

Ex:

```
data = "python"
```

```
print len(data) #6
```

“String Methods”

String Methods

- คือ วิธีการสำหรับการจัดการ string และอำนวยความสะดวก ในการเขียนโปรแกรมเป็นอย่างมาก โดยแบ่งเป็น หมวดหมู่ได้ดังนี้
 - Find and search
 - Text and sentence
 - Formatting string
 - List and sequence methods

Find and Search

- ใช้ในการค้นหาและตรวจสอบตำแหน่งของ String
 - **startswith** คือ การตรวจสอบว่ามีนั้น string เริ่มต้นด้วยคำที่เราต้องการหรือไม่
 - **endswith** คือ การตรวจสอบว่ามีนั้น string จบด้วยคำที่เราต้องการหรือไม่
 - **find** คือ การหาตำแหน่งของ string
 - **replace** คือ การแทนที่ string
 - **count** คือ การนับจำนวนของ string

Find and Search

Ex:

```
data = "python"
```

```
print data.startswith("py") #True
```

```
print data.endswith("on") #True
```

Find and Search

Ex:

```
data = "python"
```

```
print data.find("y") #1
```

```
print data.replace('py', 'qz') #qzthon
```

```
print data.count('t') #1
```

Text and Sentence

- ใช้ในการจัดแสดงข้อความให้อยู่ในลักษณะเดียวกัน
 - **capitalize** คือ การทำให้อักษรตัวแรกของประโยค เป็นตัวพิมพ์ใหญ่
 - **title** คือ การทำให้อักษรตัวแรกของแต่ละคำ เป็นตัวพิมพ์ใหญ่
 - **upper** คือ การทำให้ตัวอักษรทุกตัวเป็นพิมพ์ใหญ่
 - **lower** คือ การทำให้ตัวอักษรทุกตัวเป็นพิมพ์เล็ก

Text and Sentence

Ex:

```
data = "this is a python"
```

```
print data.capitalize() # "This is a python"
```

```
print data.title() # "This Is A Python"
```

```
print data.upper() # "THIS IS A PYTHON"
```

```
print data.lower() # "this is a python"
```

Formatting String

- เป็นการจัดรูปแบบของ String
 - **center** คือ การให้ string อยู่ตรงกลาง
 - **ljust** คือ การทำให้ string อยู่ซ้ายสุด
 - **rjust** คือ การทำให้ string อยู่ขวาสุด

Formatting String

➤ เป็นการจัดรูปแบบของ String

Ex:

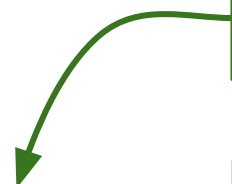
```
data= "python"
```

```
print data.rjust(10)
```

```
print data.center(10)
```

```
print data.ljust(10)
```

ขนาดของ string



python

python

python

Formatting String

Ex:

```
data= "python"
```

```
print data.center(10, '-') # --python--
```

คือ การทำให้คำอยู่ตรงกลาง และมีเครื่องหมาย -

```
print data.ljust(10, '-') # python----
```

คือ การทำให้คำอยู่ซ้ายสุด และมีเครื่องหมาย -

```
print data.rjust(10, '-') # ----python
```

คือ การทำให้คำอยู่ขวาสุด และมีเครื่องหมาย -

List and Sequence Method

- เมธอดที่ทำงานเกี่ยวกับ List
 - **join** คือ การรวม string
 - **splitlines** คือ การแยก string ออกจากการขึ้นบรรทัดใหม่
 - **split** คือ การแยก string ออกจากกันด้วย ช่องว่างหรือ string ที่เรากำหนด และนำค่าไปใส่ลงในตัวแปร list

List and Sequence Method(Join)

Ex:

```
colon = ":"
```

```
character = ["a","b","c"]
```

```
print colon.join(character) #a:b:c คือ การรวม string
```

List and Sequence Method(Splitlines)

Ex:

```
data = " " "Python is a language  
        used to create a web,  
        desktop application  
        and more" " "
```

```
data = data.splitlines()
```

```
print data
```

Output:

```
[  
    'Python is a language',  
    'used to create a web,',  
    'desktop application',  
    'and more'  
]
```

List and Sequence Method(Split)

Ex:

```
data = "This is python"
```

```
print data.split() # ['This', 'is', 'python']
```

```
data = "This is          python"
```

```
print data.split() # ['This', 'is', 'python']
```


List and Sequence Method(Split)

Ex:

```
data = "One:Two:Three:Four:Five"
```

```
print data.split(":") #['One', 'Two', 'Three', 'Four', 'Five']
```

```
print data.split(":",2) #['One', 'Two', 'Three:Four:Five']
```



ต้องการ split ที่ตำแหน่ง

Use Case

```
EX: data="""Filesystem          Size  Used Avail Use% Mounted on
      tmpfs                3.9G   0  3.9G   0% /dev/shm
      tmpfs                3.9G  26M  3.8G   1% /run
      tmpfs                3.9G   0  3.9G   0% /sys/fs/cgroup"""

print "SplitLine:",data.splitlines()
for line in data.splitlines():
    print line.split()[0],":",line.split()[1]
```

Use Case

Output:

```
SplitLine: ['Filesystem.....', '.....tmpfs.....', '.....tmpfs .....',  
'.....tmpfs.....']
```

```
Filesystem : Size
```

```
tmpfs : 3.9G
```

```
tmpfs : 3.9G
```

```
tmpfs : 3.9G
```

Exercise: String

1.แปลงข้อความใน Input ให้เป็น Output

Input = "zvzry nzw day is anothzr chancz to changz your lifz."

Output = " Every new day is another chance to change your life."

Exercise: String

2. ให้แปลงข้อความจากข้อความ Input ให้เป็นข้อความแบบ Output โดย แสดงชื่อเฉพาะ Interface ที่มี status down

Input =

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	192.168.1.3	YES	manual	down	up
GigabitEthernet2	172.17.111.3	YES	manual	up	up
GigabitEthernet3	192.168.101.3	YES	manual	up	up

Output: GigabitEthernet1 : down



“Datetime”



Content of Datetime

- Create and access datetime
- Datetime Function
- Datetime Format
- Comparison Datetime

What is Datetime?

คือ ชนิดของข้อมูลใน Python ที่เกี่ยวข้องกับวันที่ และ เวลา

How to create and access Datetime ?

- ก่อนที่จะใช้ มอดูล ตัวนี้ต้องทำการ import ก่อน โดยทำการ
- `import datetime`

Ex:

```
now = datetime.datetime.now()
new_date = datetime.datetime
(year=2018,month=1,day=1,hour=10,minute=0,second=0)

print now
print new_date #2018-01-01 10:00:00
```

“Datetime Function”

Datetime Functions

Ex:

```
now_utc = datetime.datetime.utcnow()
```

```
print now_utc
```

คือ การแสดงค่าเวลาในรูปแบบ UTC

“Datetime Format”

Datetime Format

Directive	Example	Meaning
%a	Sun	Abbreviated name of Days of Week
%A	Sunday	Full name of Days of Week
%w	0..6 (Sunday is 0)	a decimal number of Days of Week
%y	13	Year without century
%Y	2013	Year with century

Datetime Format

Directive	Example	Meaning
%b	Jan	Month as Abbreviated name
%B	January	Month as Full name
%m	01..12	Month as a decimal number
%d	01..31	Day as a zero-padded decimal number.
%e	1..31	Day as a decimal number.

Datetime Format

Directive	Example	Meaning
%H	00..23	24h Hour
%I	01..12	12h Hour
%M	00..59	Minute
%S	00..60	Second

Datetime Format

Directive	Example	Meaning
%p	AM	AM or PM
%Z	+08	Time zone
%j	001..366	Day of the year

Datetime Format

Ex:

```
date_str = "10/11/2012 13:14:15"
```

```
date = datetime.datetime.strptime  
(date_str, "%d/%m/%Y %H:%M:%S")
```

```
print date #2012-11-10 13:14:15
```

```
print type(date) #<type 'datetime.datetime'>
```

Datetime Format

Ex:

```
print date #2012-11-10 13:14:15
```

```
print type(date) #<type 'datetime.datetime'>
```

```
date = date.strftime("%Y/%m/%d %I:%M:%S %p")
```

```
print date #2012/11/10 01:14:15 PM
```

```
print type(date) #<type 'str'>
```



“Comparison
Datetime”



Comparison Datetime

Ex:

```
now = datetime.datetime.now()
```

```
print now
```

```
print datetime.timedelta(minutes=2) #0:02:00
```

```
print now-datetime.timedelta(minutes=2)
```

```
print now > datetime.datetime(2012,11,10) #True
```

```
print now < now-datetime.timedelta(minutes=2) #False
```

Comparison Datetime

Ex:

```
now = datetime.datetime.now()
```

```
last_day = now - datetime.timedelta(days=1)
```

```
print last_day
```

```
between_minutes = now - datetime.datetime(2018,5,3,10,0,0)
```

```
print between_minutes.seconds
```

Exercise: Datetime

1. จงสร้าง Function ที่รับค่าวันเกิดของตัวเองจากทางหน้าจอและคำนวณว่าตัวเองเกิดมาแล้วกี่วัน

Output:

My Birthday is

Day:18

Month:02

Year:1996

8320 days, 14:25:38.406000

Exercise: Datetime

2. ให้เขียนโปรแกรมที่สามารถ แสดงค่าวันย้อนหลังจากวันนี้ไป 7 วัน

Output:

```
วันที่ Wednesday,28-Nov-2018 02:17:12 PM
วันที่ Tuesday,27-Nov-2018 02:17:12 PM
วันที่ Monday,26-Nov-2018 02:17:12 PM
วันที่ Sunday,25-Nov-2018 02:17:12 PM
วันที่ Saturday,24-Nov-2018 02:17:12 PM
วันที่ Friday,23-Nov-2018 02:17:12 PM
วันที่ Thursday,22-Nov-2018 02:17:12 PM
```



“Try Except”



Content of Try except

- Try except Format
- Errors and Exceptions
- Raising Exception

What is error handling ?

คือ กลไกในการจัดการความผิดปกติของโปรแกรม

What is Try Except?

คือ คำสั่งที่เป็น error handling ของภาษา python



“Try Except Format”



Try Except Format

try:

TAB คำสั่ง

การลองทำโปรแกรมที่เราออกแบบ

except [expression [, target]]:

TAB คำสั่ง

จะทำงานเมื่อเกิด Error

ชนิดของ Error เช่น Type Error ซึ่ง
สามารถใส่ค่าหรือไม่ก็ได้

เราสามารถนำ Error ที่เกิดไปเก็บใน
ตัวแปรได้ เช่น Exception as err

Try Except Format

try:

TAB คำสั่ง

except [expression [, target]]:

TAB คำสั่ง

else:

TAB คำสั่ง

finally:

TAB คำสั่ง

ทำงานก็ต่อเมื่อโปรแกรมไม่
เกิด Error

ทำงานเสมอไม่ว่าโปรแกรม
จะเกิด Error หรือไม่

Try Except Format

Ex:

```
while True:
```

```
    a = float(input("Enter first number: "))
```

```
    b = float(input("Enter second number: "))
```

```
    print "%d / %d =  %f" % (a, b, a / b)
```

Try Except Format

Output:

Enter first number: test

```
a = float(input('Enter first number: '))  
File "<string>", line 1, in <module>  
NameError: name 'test' is not defined
```


Try Except Format

Ex: while True:

try:

a = float(input("Enter first number: "))

b = float(input("Enter second number: "))

print "%d / %d = %f" % (a, b, a / b)

except:

print "Invalid Input"

Try Except Format

Output:

Enter first number: test

Invalid Input

Enter first number: 10

Enter second number: 0

Invalid Input

“Error and Exception”

Errors and Exceptions

ข้อผิดพลาดในภาษา Python ในการเขียนโปรแกรมมีอยู่ 2 ชนิด

1. Syntax Errors
2. Exceptions

The word "Error" is displayed in a large, bold, blue font. The letters are filled with a blue color and have a white outline. The background of the letters is a white monospace font, which is a common style for code or technical text. The word is centered horizontally and occupies a significant portion of the lower half of the slide.

Errors and Exceptions

1. Syntax Errors

เป็นการเขียนผิดหลักไวยากรณ์ของภาษา Python มักจะพบบ่อย เมื่อลืมใส่วงเล็บหรือใช้คำสั่งผิด import ไลบรารีผิด หรือนำโค้ด Python 2 มารันบน Python 3

Errors and Exceptions

1. Syntax Errors

Ex:

```
while True print 'Hello world'
```

```
File "<stdin>", line 1
```

```
while True print 'Hello world'
```

```
      ^
```

```
SyntaxError: invalid syntax
```

Errors and Exceptions

2. Exceptions

แม้ว่าจะเขียนโค้ดโปรแกรมถูกหลักไวยากรณ์ของภาษา Python แต่หากจะมีปัญหาในการดำเนินการทำงานของโปรแกรม เช่น ใช้ชนิดของตัวแปรผิด

- **IOError** : ไม่สามารถเปิดไฟล์ได้
- **ImportError** : ไม่พบไลบรารีที่ระบุไว้
- **TypeError** : ใช้ชนิดของตัวแปรผิด
- **ZeroDivisionError** : เป็นข้อผิดพลาดที่พบเมื่อคุณนำค่าจำนวนเต็มหรือจำนวนจริงหารด้วย 0 เพราะการหารด้วย 0 ไม่สามารถหารได้

Errors and Exceptions

2. Exceptions

Ex:

```
print 10 / 0  
print 5 * money
```

Output:

Traceback (most recent call last):

File "exception.py", line 1, in <module>
print (10 / 0)

ZeroDivisionError: division by zero

Traceback (most recent call last):

File "exception.py", line 3, in <module>
print (5 * money)

NameError: name 'money' is not defined

Errors and Exceptions

2. Exceptions

Ex:

```
try:
```

```
    cal = 10/0
```

```
except ZeroDivisionError:
```

```
    print("เกิดข้อผิดพลาดในการแปล")
```

Output:

เกิดข้อผิดพลาดในการแปล



“Raising Exception”



Raising Exception

โปรแกรมเมอร์สามารถสั่งให้เกิด Exception ขึ้นเองได้ โดยการใช้ คำสั่ง raise

Raising Exception

Ex: try:

```
    name = input("Enter your name: ")
    if name == 'mateo':
        raise Exception("Whoa! Mateo you are not allowed here")
    print "Hi ", name
except Exception as err:
    print "Exception: ", err
else:
    print "Welcome"
finally:
    print "===END==="
```

Raising Exception

Output:

Enter your name: mateo

Exception: Whoa! Mateo you are not allowed here

===END===

Enter your name: Marcus

Hi Marcus

Welcome

===END===

Use Case

Ex:

try:

my_db.ConnectDB() #Fuction ที่ใช้ในการconnect database

except Exception as e:

print "connect database error:" + str(e)

Use Case

Output:

```
connect database error:('Unable to connect to any servers',  
{'127.0.0.1': error(10061, "Tried connecting to [('127.0.0.1',  
9042)]. Last error: No connection could be made because the  
target machine actively refused it")}))
```

Exercise: Try except

1.สร้างโปรแกรมที่คำนวณ หาความหนาแน่นโดยใช้สูตรด้านล่าง โดยหากใส่ 0 หรือ ตัวหนังสือ โปรแกรมก็ยังสามารถทำงานต่อได้

$$\text{ความหนาแน่น} = \frac{\text{มวล}}{\text{ปริมาตร}}$$

HINT: `ZeroDivisionError` :ไม่สามารถหารด้วย 0 ได้
`TypeError`:ใช้ตัวแปรผิดชนิด

Output:

มวล:50

ปริมาตร:0

ไม่สามารถหารด้วย 0 ได้

ขอบคุณค่ะ

มวล:'test'

ปริมาตร:'test'

ไม่สามารถใส่ตัวอักษรได้

ขอบคุณค่ะ

มวล:50

ปริมาตร:5

ความหนาแน่น มีค่าเท่ากับ 10

การหาความหนาแน่นเสร็จสิ้น

THANK YOU

contact:

sdi-inet@inet.co.th

