TIN HỌC CƠ SỞ

Phần 2: Ngôn ngữ lập trình Python

CHƯƠNG 10.

HÀM XỬ LÝ NUMBER STRING DATE TIME

Tin cơ sở (LT): 010100229802 - DHKL16A1HN, - DHKL16A2HN

GV. Cao Diệp Thắng

Mục tiêu chương

Nắm vững:

- ❑ Cú pháp một số hàm/phương thức xử lý các kiểu dữ liệu Number, string,
- ❑ Cú pháp, cách sử dụng random(), hàm xử lý thời gian

Nội dung

1. Thư viện một số hàm toán học
2. Số ngẫu nhiên
3. phương thức xử lý chuỗi ký tự trong python
4. phương thức/hàm xử lý thời gian

10.1. NUMBER

10.1.1.Thư viện hàm toán học trong python

❑ *Python cung cấp sẵn thư viện **math** với các phương thức/hàm toán học*

module	Hàm	Kết quả trả về	Ví dụ
import math	abs(x)	|x|	>> x =2, >>print(abs(x)) kết quả : 2
	ceil(x)	▪ Giá trị nguyên nhỏ nhất lớn hơn x	>>print(math.ceil(5.5)) 6
	exp(x)	e^x	>>print(math.exp(4)) 20.085536923187668
	fabs(x)	|x|	>>print(math.fabs(-9)) 9.0
	floor(x)	▪ Giá trị nguyên lớn nhất nhỏ hơn x	>>print(math.floor(6.25)) 6
	log(x)	log(x), x>0	>>print(math.log(5)) 1.6094379124341003
	max(x1,x2,…)	▪ Giá trị lớn nhất của x1,x2,..	>>print(max(20,52,15)) 52
	min(x1,x2,…)	▪ Giá trị nhỏ nhất của x1,x2,..	>>> print(min(23,5,15)) 5

	mod(x)	▪ Tách x thành hai phần:
		Phần lẻ và phần nguyên, kết quả trả về đều là kiểu float
			>>> print(math.modf(3.5))
			(0.5, 3.0)
			>>> print(math.modf(10))
			(0.0, 10.0)

	pow(x,y)	x^y	>>> print(math.pow(2,3)) 8.0
	round(x1[,n])	Làm tròn, với n số lẻ	>>> print(round(12.345672,3)) 12.346
	sqrt(x)	Căn bậc 2 của x	>>> print(math.sqrt(8)) 2.8284271247461903

10.1.2. Tạo số ngẫu nhiên random

a. Ngẫu nhiên thực và giả ngẫu nhiên

- Số ngẫu nhiên có thể thu được do áp dụng các phương pháp toán học hay được gọi là bộ tạo số ngẫu nhiên (Random Number Generator - RNG). Nó có thể được chia thành 2 loại: *bộ tạo số ngẫu nhiên thực* (True Random Number Generators - TRNGs hay còn gọi là bộ tạo số ngẫu nhiên phần cứng) và *bộ tạo số giả ngẫu nhiên* (Pseudo-random Number Generator - PRNGS).

b. Bộ tạo số ngẫu nhiên thực – TRNG (True Random Number Generators)

Các bộ tạo số ngẫu nhiên thực là các phương pháp trích rút tính ngẫu nhiên hoặc không thể tiên đoán từ các khía cạnh không thể đoán trước được của các tiến trình vật lý. Các phương thức này không trực tiếp tạo ra các số, mà là các trạng thái, sau đó có thể được diễn dịch sang dạng số - đây là lý do tại sao chúng thường được gọi là các trình tạo sự kiện ngẫu nhiên (Random Event Generators - REGs). Một số trong số chúng, sử dụng các sự kiện vĩ mô phổ biến, như là các phương pháp ném xúc xắc, lật đồng xu hoặc xáo trộn thẻ bài.

c. Bộ tạo số giả ngẫu nhiên – PRNG (Pseudo Random Number Generators)

Khái niệm: PRNG là các thuật toán, sử dụng một phần nhỏ thông tin (được gọi là **seed**) và sau đó áp dụng các công thức toán học phức tạp để tạo ra các bộ số xác định giống như các bộ thực sự ngẫu nhiên. **Seed** có thể là một giá trị được lấy từ một trình tạo số ngẫu nhiên thực sự hoặc một nguồn khác, như đồng hồ của hệ thống hoặc thời gian hiện tại.

- Chạy bộ tạo số nhiều lần bằng *cùng một seed* → *cùng một output* mỗi lần chạy.
- Các số kết quả hầu như không thể phân biệt được với các số có nguồn gốc từ các bộ tạo số ngẫu nhiên thực, mặc dù thực tế có một số quy tắc ẩn trong sự phân phối của chúng.
- Đối với nhiều ứng dụng, loại giả ngẫu nhiên xác định này là hoàn toàn đủ đáp ứng yêu cầu.

d. Module Random trong Python

- Python cung cấp sẵn một module cực kỳ dễ sử dụng để xử lý với các số ngẫu nhiên.
- Module này gọi là random, được cài đặt một bộ tạo số giả ngẫu nhiên và chứa các hàm cho phép giải quyết trực tiếp nhiều vấn đề lập trình khác nhau sử dụng đến tính ngẫu nhiên.
- Module random dựa trên Marsenne Twister - một thuật toán rất phổ biến, là trình tạo số giả ngẫu nhiên mặc định không chỉ cho Python, mà còn cho nhiều hệ thống phần mềm phổ biến khác như Microsoft Excel, MATLAB, R hay PHP.
- Ưu điểm nổi bật của nó là việc được cấp phép chứng nhận, tính ngẫu nhiên được xác nhận bởi nhiều thử nghiệm thống kê và tốc độ tương đối cao so với các PRNG khác.

a. Hàm random()

*Phương thức quan trọng nhất của module random là phương thức **random()**. Hầu hết các chức năng khác phụ thuộc vào nó. Phương thức random() tạo ra một số thực float ngẫu nhiên trong phạm vi **(0.0, 1.0).***

Ví dụ 10.1

```
>>> import random
>>> random.random()
0.7266574065857312
>>> random.random()
0.9034837449669634
>>> random.random()
0.19915184214690296
```

Lưu ý: *không giống như tung một đồng xu, mô-đun random tạo ra các con số giả ngẫu nhiên hoàn toàn xác định, vì vậy nó không thích hợp cho các mục đích mật mã.*

a. Hàm seek()

- Nếu chúng ta không đặt một seed cho bộ tạo số giả ngẫu nhiên, thì *seed* mặc định là thời gian hệ thống hiện tại. Tuy nhiên, chúng ta có thể đặt giá trị chính xác của *seed* một cách thủ công, việc này sẽ rất hữu ích nếu chúng ta muốn sao chép kết quả giả ngẫu nhiên trong tương lai. Với mục đích như vậy, chúng ta có thể sử dụng phương thức **random.seed**()

  Phương thức **random.seed()** sẽ ảnh hưởng đến tất cả các phương thức của module random mà chúng ta sử dụng sau khi gọi nó. Trong đoạn code ví dụ 10.2, đặt seed là 5 và sau đó gọi hàm random.random() nhiều lần

Ví dụ 10.2

```
>>> import random
>>> random.seed(5)
>>> random.random()
0.6229016948897019
>>> random.random()
0.7417869892607294
>>> random.random()
0.7951935655656966
```

```
>>> random.seed(5)
>>> random.random()
0.6229016948897019
>>> random.seed(7)
>>> random.random()
0.3238327648331623 7
```

d. Module Random trong Python,…

Lưu ý :

- **seed** do người dùng định nghĩa sẽ chỉ được sử dụng lần đầu tiên khi có một phương thức random khác được thực thi - sau đó, các seed cho các phương thức sau sẽ thay đổi bằng cách sử dụng các giá trị ngẫu nhiên được tạo ra trước đó.
- **seed** cho phép Python đưa ra được giá trị ngẫu nhiên mới mỗi lần. Tuy nhiên, sau khi thiết lập lại seed bằng phương thức random.seed(), chúng ta sẽ có thể sao chép chính xác chuỗi số giả ngẫu nhiên bất cứ lúc nào.
- Nếu người dùng đưa ra cùng một **seed** mỗi khi họ chạy một thử nghiệm có sử dụng một trong các phương pháp random thì người dùng vẫn có thể biết đầu ra sẽ là gì cho các thử nghiệm này.

Hàm randint() lấy hai đối số thể hiện phạm vi mà phương thức rút ra một số nguyên ngẫu nhiên.

Ví dụ 10.3

```
>>> random.randint(1,10)
3
>>> random.randint(1,10)
7
>>> random.randint(1,10)
1
```

Bảng 10.2. Tạo số ngẫu nhiên random trong python

module	Phương thức/hàm	Kết quả trả về	Ví dụ

import random	randrange([start,] stop [,step])	▪ randrange(x) để tạo ra một số nguyên ngẫu nhiên nhỏ hơn x
		▪ randrange(a, b[,step]) để tạo một số ngẫu nhiên từ range(a, b, step)
			>>> random.randrange(100)
			49
			>>> random.randrange(100)
			2
			>>> random.randrange(0,100,3)
			36
			>>> random.randrange(0,100,3)
			87
			>>> random.randrange(1,6)
			5
			>>> random.randrange(1,6)
			3

	random()	*Chú ý: Để sử dụng hàm random() cần import module random*	>>> print(random.random()) 0.563014574950693 >>> print(random.random()) 0.573003525695827 1 >>
	uniform(x,y)	Một số thực ngẫu nhiên trong khoảng >x và <y	

10.2. STRING Một số phương thức xử lý chuỗi ký tự trong python

tt	Phương thức	Kết quả trả về	Ví dụ

1	**capitalize**()	Bản sao của chuỗi với ký tự đầu tiên viết hoa.	>>> str = "thang long"
			>>> print(str.capitalize())
			Thang long

2	**center**(with[,fillchar])	Bản sao của chuỗi bằng cách di chuyển chuỗi ra giữa và điền vào đầu và cuối chuỗi ký tự fillchar, mặc định sẽ tự đồng điền khoảng trắng.	>>> str = "Happy birthday"
			>>> print(str.center(30,"*"))
			**********Happy birthday**********
			>>> print(str.center(30,"-"))
			----------Happy birthday----------
			>>> print(str.center(30,"o"))
			ooooooooHappy birthdayoooooooo
			>>> print(str.center(30))
			Happy birthday

3	**count**(sub,[start,end)	Số lần chuỗi con sub xuất hiện trong chuỗi tính từ start đến end	>>> str="Happy new year 2022"
			>>> print(str.count("p",0,10))
			2
			>>> print(str.count("new",0,10))
			1

4	**find**(str2,[beg,end])	Tìm chuỗi str2 từ vị trí beg đến vị trí end trong chuỗi và trả về vị trí nếu tìm thấy, -1 nếu không tìm thấy, mặc định beg=0 và end=len	>>> str1="Happy new year"
			>>> str2="new"
			>>> print(str1.find(str2))
			6
			>>> print(str1.find(str2,3,10))
			6
			>>> print(str1.find(str2,15))
			-1

Một số phương thức xử lý chuỗi ký tự trong python,...

tt	Phương thức	Kết quả trả về	Ví dụ

5	**isdigit**()	Nếu chuỗi chỉ chứa các ký tự số thi trả về giá trị True, ngược lại trả về False	>>> str="123456"
			>>> print(str.isdigit())
			True
			>>>
			>>> str="123a456"
			>>> print(str.isdigit())
			False

6	**replace**(old,new[,max])	Bản sao của chuỗi với old được thay thế bằng new, nếu có max thì chỉ thay thế một lấn	str="She is beautiful. He is handsome."
			>>> print(str.replace("is", "was"))
			She was beautiful. He was handsome.
			>>> print(str.replace("is", "was", 1))
			She was beautiful. He is handsome.

7	**strip**([chars])	Bản sao của chuỗi sau khi đã loại bỏ **chars** ở đầu và cuối chuỗi, mặc định là loại bỏ khoảng trắng	>>> str = " Happy new year "
			>>> print(str.strip())
			Happy new year
			>>>
			>>> str="aaaHAPPY NEW YEARaaaaa"
			>>> print(str.strip("a"))
			HAPPY NEW YEAR

Một số phương thức xử lý chuỗi ký tự trong python,...

tt	Phương thức	Kết quả trả về	Ví dụ

8	split(str[, num])	Từ chuỗi gốc, tạo ra list các phần tử được cắt ra theo str, mặc định str là khoảng trắng; nếu thiết lập num thì sẽ qui định num+1 số chuỗi con được tạo ra	str="Python is programming Language"
			>>> print(str.split())
			['Python', 'is', 'programming','Language']
			>>> print(str.split("o"))
			['Pyth', 'n is pr', 'gramming Language']
			>>> print(str.split(" ",2))
			['Python', 'is', 'programming Language']
			>>> print(str.split(" ",1))
			['Python', 'is programming Language']

10.3. DATETIMES

Python cung cấp sẵn một số phương thức/hàm xử lý thời gian (**import time**, **import datetime**). **Chú thích**: Đối với hệ thống Unix, 00:00:00 ngày 1/1/1970 theo giờ UTC được gọi là **epoch** (thời điểm bắt đầu thời gian). UTC là một tiêu chuẩn quốc tế được sử dụng để thiết lập cho tất cả múi giờ trên thế giới.

- **time**.time()

trả về số giây tính từ *epoch*, hay còn gọi là giá trị *timestamp*

Ví dụ 10.4

```
import time
seconds = time.time()
print("So giay tinh tu epoch:", seconds)
```

Kết quả:

```
So giay tinh tu epoch:  1658202589.192107
```

❑ time.localtime()

Hàm localtime() trong module time lấy số giây được truyền vào làm đối số và trả về struct_time theo giờ địa phương.

Ví dụ 10.5

```
>>> import time
>>> result = result = time.localtime(time.time())
>>> print("Ket qua:", result)
Ket qua: time.struct_time(tm_year=2022, tm_mon=7, tm_mday=19, tm_hour=10,
tm_min=57, tm_sec=32, tm_wday=1, tm_yday=200, tm_isdst=0)
>>> print("\nNam:", result.tm_year)

Nam: 2022
>>> print("Gio:", result.tm_hour)
Gio: 10
```

Bảng 10.3. Giá trị các thuộc tính của struc_time

tt	Thuộc tính	Giá trị
0	**tm_year**	trả về năm hiên tại, vd: 2022
1	**tm_mon**	1 đến 12
2	**tm_mday**	1 đến 31
3	**tm_hour**	0 đến 23
4	**tm_min**	0 đến 59
5	**tm_sec**	0 đến 61 (60 hoặc 61 là giây nhuận)
6	**tm_wday**	0 đến 6 (0 là Thứ Hai)
7	**tm_yday**	1 đến 366 (ngày Julian)
8	**tm_isdst**	-1, 0, 1, -1 nghĩa là thư viện xác định DST DST: "Daylight Saving Time" (Giờ tiết kiệm ánh sáng ban ngày) [17], [*]

* https://bom.so/GWhtoJ

❑ time.ctime()

Phương thức này chuyển đổi một time được biểu diễn bằng số giây tính từ *epoch* thành một biểu diễn ở dạng chuỗi.

Ví dụ 10.6

```
1    import time
2    #Số giây tính từ epoch
3    #edited bởi khoa khoa học dữ liệu Uneti
4    seconds = time.time()
5    local_time = time.ctime(seconds)
6    print("Loacl time: ", local_time)
```

Kết quả:

```
Loacl time:  Tue Jul 19 22:37 2022
```

❑ time.sleep()

Hàm **sleep**() dừng thực thi luồng hiện tại trong số giây truyền vào.

Ví dụ 10.7

```
1    import time
2
3    print("Start : ", time.ctime())
4    time.sleep(3)
5    print("End :", time.ctime())
```

Kết quả:

```
Start :  Tue Oct 11 11:48:30 2022
End : Tue Oct 11 11:48:33 2022
```

❑ time.asctime()

Trả về chuỗi thời gian theo định dạng

Ví dụ 10.8

```
1    import time
2
3    localtime = time.asctime(time.localtime(time.time()))
4    print("Thời gian cục bộ hiện thời: \n", localtime)
```

Kết quả:

```
Thời gian cục bộ hiện thời:
Tue Oct 11 11:51:24 2022
```

```
2022-30-11 ……
```

❑ datetime.now()

Trả về ngày giờ hiện thời của hệ thống

Ví dụ 10.9

```
1    #datetime.now phương thức/hàm trả về ngày giờ hệ thống
2    from datetime import datetime
3    print(datetime.now())
```

Kết quả: trả về ngày giờ hệ thống

```
>>> from datetime import datetime
>>> print(datetime.now())
2022-11-30 07:00:26.405954
```

❑ Datetime(year, month, day)

Trả về một ngày dựa trên **year**, **month**, **day**, nếu hợp lệ; ngược lại sẽ phát sinh ra lỗi

Ví dụ 10.10

```
>>> from datetime import datetime
>>> print(datetime(2022, 7, 22))
2022-07-22 00:00:00
```

```
>>> print(datetime(2022, 07, 22))
SyntaxError: invalid token
>>> print(datetime(2022, 17, 22))
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print(datetime(2022, 17, 22))
ValueError: month must be in 1..12
```

❑ `date_variable.strftime(`chuỗi định dạng mẫu`)`

Hiển thị ngày tháng theo chuỗi định dạng mẫu

Trong đó:

%Y:	năm	[0001,..., 2018, 2019,..., 9999]
%m:	tháng	[01, 02, ..., 11, 12]
%d:	ngày	[01, 02, ..., 30, 31]
%H:	giờ	[00, 01, ..., 22, 23]
%M:	tháng	[00, 01, ..., 58, 59]
%S:	giây	[00, 01, ..., 58, 61]

Ví dụ 10.11

```
1   from datetime import datetime
2   ngay_hien_tai = datetime.now()
3   print('Ngày tháng năm : ', ngay_hien_tai.strftime("%d-%m-%Y"))
```

```
Ngay thang nam : 30-11-2022
```

Làm việc với calendar (import calendar)

- **`calenda.isleap(year)`**

Trả về **True** nếu là năm nhuận, ngược lại trả về **False**

Ví dụ 10.12

```
>>> import calendar
>>> print(calendar.isleap(2022))
False
>>>
>>> print(calendar.isleap(2024))
True
```

Làm việc với calendar (import calendar)

- **calendar.monthcalendar(year, month)**

  Trả về danh sách các ngày trong tháng, năm được chọn, chia theo từng tuần, với giá trị 0 là ngày không nằm trong tháng của năm.

Ví dụ 10.13

```
>>> import calendar
>>> print(calendar.monthcalendar(2022,12))
[[0, 0, 0, 1, 2, 3, 4], [5, 6, 7, 8, 9, 10, 11], [12, 13, 14, 15, 16, 17, 18]
, [19, 20, 21, 22, 23, 24, 25], [26, 27, 28, 29, 30, 31, 0]]
```

Làm việc với calendar (import calendar)

- **`calendar.weekday(`**year,month,day**`)`**

  Trả về thứ của ngày/tháng/năm, với giá trị số **0**: Monday, **1**: Tuesday,…

Ví dụ 10.14

```
>>> import calendar
>>> print(calendar.weekday(2022,12,31))
5
>>> print(calendar.weekday(2022,11,28))
0
>>> print(calendar.weekday(2022,11,29))
1
>>> print(calendar.weekday(2022,11,30))
2
```

Làm việc với calendar (import calendar)

❑ `calendar.monthrange(year,month)`

Trả về hai phần tử:

- Phần tử thứ nhất là thứ của ngày đầu tiên trong tháng,
- Phần tử thứ hai là số ngày trong tháng.

Ví dụ 10.15

```
>>> import calendar
>>> print(calendar.monthrange(2022,10))
(5, 31)
>>> print(calendar.monthrange(2022,11))
(1, 30)
>>> print(calendar.monthrange(2022,12))
(3, 31)
```

Câu hỏi thảo luận

1. Nêu khái niệm thư viện hàm toán học trong Python?
2. Trình bày khái niệm ngẫu nhiên? Bộ tạo số ngẫu nhiên trong Python?
3. Khái niệm **epoch** time là gì?
4. Để xử lý chuẩn hóa chuỗi ký tự, có thể dùng các hàm nào trong thư viện Python?
5. giữa thư viện time và datetime?

Bài tập về nhà:

Bài tập chương 10, TLHT