제 15 강 Time & Thread

학습 목차

- 시간 측정
- 날짜와 시각 처리
- 멀티쓰레딩
- 서브프로세스 – 외부 프로그램 실행

시간 및 날짜 관련 라이브러리

- time – 시간 액세스와 변환
- datetime – 기본 날짜와 시간형
- dateutil – 시간대 및 구문 분석 지원
- calendar – 달력 관련 함수

Time 모듈

- Unix epoc time
  - 1970년 1월 1일 AM 0시 기준.
  - 많은 프로그램들의 기준 시작 시간으로 활용됨.

- Time.time()
  - Unix epoc time에서 시작되어 경과된 시간(초)

```
>>> import time
>>> time.time()
1621468685.019336
```

실행 시간 측정

```
def calc():
    global prod
    prod = 1
    for i in range(1,100000000):
        prod += i

start_time = time.time()
calc()
end_time = time.time()

print(f'{prod=} in {end_time-start_time} sec')
```

본격적인 프로파일링

```
# 정확한 결과
import timeit
timeit.timeit(calc, number=1) # number 기본값은 백만번..수행 후, 평균값을 구함.

# 프로그램 전체에 대한 통계적인 데이터, 부하가 추가됨. (실제 실행 시간이 늘어남)
import cProfile
cProfile.run('calc()')
```

time functions

```
time.gmtime()                           # 현재 시각, UTC 기준
time.gmtime(0)                          # EPOC time
time.gmtime(1000000000)
time.localtime()                        # 현재 지역 시각
time.localtime(0)                       # 현재 지역에서 0초가 경과했을 때,

time.ctime()
time.ctime(0)

time.timezone                           # 9 hours * 60 min * 60 secs
time.tzname
```

datetime 모듈

- 날짜 요일 등을 효과적으로 처리하는 모듈
- 날짜 사칙 연산, 변환 등등

datetime

```
import datetime
datetime.datetime.now()
datetime.date.today()
target_time = datetime.datetime(2070, 8, 22, 13, 10)
print(target_time)
str(target_time)

dt = datetime.datetime.now()
dt.day
dt.year
dt.time()
dt.hour
dt.minute
dt.microsecond

datetime.datetime.fromtimestamp(100000)
datetime.datetime.fromtimestamp(time.time())
```

날짜 비교

```
today = datetime.date.today()
today.year, today.month, today.day
birthday = datetime.date(today.year, 8, 22)
today > birthday
today < birthday
```

시간 차이

```
# time delta
delta = birthday - today
delta
delta = datetime.timedelta(days=10, hours=1, minutes=3)
delta.total_seconds()
delta
print(delta)
str(delta)
```

```
# next 1000 days later
thousand_days = datetime.timedelta(days=1000)
today = datetime.date.today()
thousand_days_later = today + thousand_days
thousand_days_later
thousand_days_later - today
```

Time alarm

```
import datetime
import time
halloween_datetime = datetime.datetime(datetime.date.today().year, 10, 31)
while datetime.datetime.now() < halloween_datetime:
    time.sleep(1)
print('THIS IS HALLOWEEN')
```

datetime object formatting

Table 15-1: `strftime()` Directives

strftime directive	Meaning
%Y	Year with century, as in '2014'
%y	Year without century, '00' to '99' (1970 to 2069)
%m	Month as a decimal number, '01' to '12'
%B	Full month name, as in 'November'
%b	Abbreviated month name, as in 'Nov'
%d	Day of the month, '01' to '31'
%j	Day of the year, '001' to '366'
%w	Day of the week, '0' (Sunday) to '6' (Saturday)
%A	Full weekday name, as in 'Monday'
%a	Abbreviated weekday name, as in 'Mon'
%H	Hour (24-hour clock), '00' to '23'
%I	Hour (12-hour clock), '01' to '12'
%M	Minute, '00' to '59'
%S	Second, '00' to '59'
%p	'AM' or 'PM'
%%	Literal '%' character

strftime, strptime

```
# datetime 오브젝트를 문자열로 변환
now = datetime.datetime.now()
now.strftime('%Y/%m/%d %H:%M:%S')
now.strftime('%I:%M %p')
now.strftime("%B of '%y")

# 문자열을 datatime 오브젝트로 변환
datetime.datetime.strptime('August 22, 1970', '%B %d, %Y')
datetime.datetime.strptime('2021/08/22 13:29:00', '%Y/%m/%d %H:%M:%S')
datetime.datetime.strptime("August of '70", "%B of '%y")
```

시간대 처리

```
import pytz


# list up supported named timezones
for tz in pytz.all_timezones:
    print(tz)

UTC = datetime.timezone.utc   # datetime provides UTC by itself
UTC = pytz.timezone('UTC') # same as above
KST = pytz.timezone('Asia/Seoul') # pytz provides named timezone info

now_kst = datetime.datetime.now(KST)    # current time as KST
now_utc = now_ktc.astimezone(UTC)       # convert to UTC

now_utc - now_kst  # equals timedelta(0)
```

시간대	나라	코드
UTC-5	미국(동부)	EST
UTC	영국	GMT
UTC+8	대만	TW
UTC+9	대한민국	KST
UTC+9	일본	JST
UTC+10	오스트레일리아(동부)	AEST

calendar

https://docs.python.org/ko/3/library/calendar.html

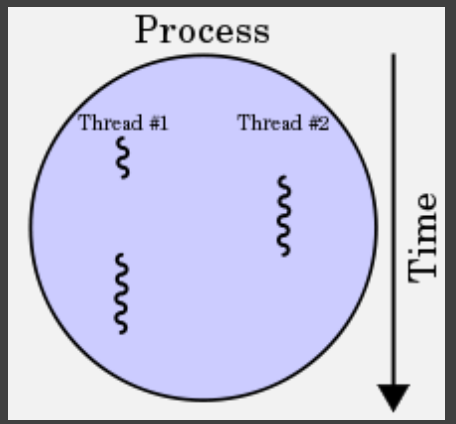
```
import calendar

this_year = datetime.datetime.now().year

calendar.calendar(this_year)
print(calendar.calendar(this_year))
calendar.setfirstweekday(0)
print(calendar.calendar(this_year))
print(calendar.month(this_year,5))

calendar.weekday(this_year, 5, 25)
calendar.monthcalendar(this_year, 5)
```

calendar 오브젝트

```
cal = calendar.Calendar(firstweekday=6)

for date in cal.itermonthdates(this_year, 6):
    print(date)

for day in cal.itermonthdays(this_year, 6):
    print(day)

for day in cal.itermonthdays2(this_year, 6):
    print(day)

for day in cal.itermonthdays3(this_year, 6):
    print(day)

for day in cal.itermonthdays4(this_year, 6):
    print(day)

cal.monthdatescalendar(this_year, 6)
cal.monthdayscalendar(this_year, 6)
cal.monthdays2calendar(this_year, 6)
```

Multithreading

- Thread – 독립적인 프로그램 실행 흐름
- Multi-threading – 여러 개의 thread를 동시에 실행하는 기능
- Thread vs Process
  - 프로세스가 더 큰 개념 – thread is a subset of a process
  - Thread는 동일한 메모리 공간 공유. Process (X)
- Concurrency issue에 대한 해결책 구현 필요.
  - 변수를 동시에 읽고, 쓰게 되는 경우.



Thread object 생성 및 run

```
import threading

looping = True

def calc(number=100000, show='*'):
    global looping
    looping = True
    prod = 1
    for i in range(1, number):
        if i % (number//10) == 0:
            print(f'{show*3}{i//(number//10)}', end='')
        prod *= i
    print('\nCalculation Finished.')
    looping = False

thread = threading.Thread(target=calc)
thread.start()
print('Program ends, but looping is', looping)
```

인자 전달

```
thread = threading.Thread(target=calc, args=[200000])
thread.start()
print('Program ends, but looping is', looping)

thread = threading.Thread(target=calc, args=[100000, '+'])
thread.start()
print('Program ends, but looping is', looping)

thread = threading.Thread(target=calc, args=[100000], kwargs={'show':'='})
thread.start()
print('Program ends, but looping is', looping)
```

Thread 완료 대기

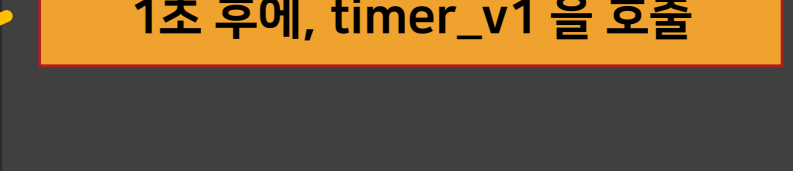
```
thread = threading.Thread(target=calc)
thread.start()
print('looping is', looping)
thread.join()
print('looping is', looping)
print('Program ends')
```

Thread 종료

- thread._stop() 이 있으나, 이것을 쓰는 것 보다는, looping 변수 등을 이용해서, 중단시키고, join()으로 마무리하는 것이 타당..

Timer 구현 #1.

```
import threading
import datetime

prev_callback_time = datetime.datetime.now()
def callback():
    global prev_callback_time
    now = datetime.datetime.now()
    delta = now - prev_callback_time
    prev_callback_time = now
    print(f'callback @{now} ({delta})')

looping = True

def simple_timer():
    callback()
    if looping:
        threading.Timer(1, simple_timer).start()

simple_timer()
```



Timer 구현 #2.

```
def simple_timer():
    global prev_now

    while looping:
        now = datetime.datetime.now()
        if now >= prev_now + datetime.timedelta(seconds=1):
            callback()
            prev_now = now

thread = threading.Thread(target=simple_timer)
thread.start()
```

Timer 구현 #3.

```
def simple_timer():
    global prev_now

    while looping:
        now = datetime.datetime.now()
        if now.second != prev_now.second:
            callback()
            prev_now = now
```

Timer 구현 #4.

```
def simple_timer():
    global prev_now

    prev_delta = datetime.timedelta(0)
    while looping:
        now = datetime.datetime.now()
        delta = now - prev_now + prev_delta

        if delta >= datetime.timedelta(seconds=1):
            callback()
            prev_delta = delta - datetime.timedelta(seconds=1)
            prev_now = now
```

Tkinter after()

```
import tkinter as tk
import time

class SampleApp(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        self.clock = tk.Label(self, text="")
        self.clock.pack()

        # start the clock "ticking"
        self.update_clock()

    def update_clock(self):
        now = time.strftime("%H:%M:%S" , time.localtime())
        self.clock.configure(text=now)
        # call this function again in one second
        self.after(1000, self.update_clock)

if __name__== "__main__":
    app = SampleApp()
    app.mainloop()
```

외부 프로그램의 실행 subprocess

```
import subprocess
import sys

subprocess.run('c:/Windows/System32/calc.exe')

subprocess.run(['c:/Windows/notepad.exe', 'simple.txt'])

subprocess.run([sys.executable, 'hello.py'])

subprocess.run(['start', 'kitty.jpg'], shell=True)

subprocess.run(['start', 'alarm.wav'], shell=True)
```