

LESSON

Cron을 활용한 스케줄링

```
animLength = toTime - fromTime

# Ask user for directory
filePath = c4d.storage.SaveDialog()
filePath, objName = os.path.split(filePath)
objName = objName + "_"
filePath = filePath + "\\ "

# Ask for confirmation
questionDialogText = "Obj Sequence will be saved as:\n\n" \
    "" + filePath + objName + "###.obj\n\n" \
    "From frame " + str(fromTime) + " to " + str(toTime) + " "
proceedBool = c4d.gui.QuestionDialog(questionDialogText)

if proceedBool == True:

    # Loop through animation and export frames
    for x in range(0,animLength):

        # change frame, redraw view
        moveTime = c4d.BaseTime(fromTime,docFps) + c4d.BaseTime(x,docFps)
        doc.SetTime(moveTime)
        c4d.EventAdd(c4d.EVENT_FORCEREDRAW)
        c4d.DrawViews(c4d.DRAWFLAGS_FORCEFULLREDRAW)

        # update status bar
        c4d.StatusSetText("Exporting " + str(x) + " of " + str(animLength))
        c4d.StatusSetBar(100.0*x/animLength)

        # add buffer 0.001
        bufferedNumber = str(doc.GetTime().GetFrame(docFps))
        if len(bufferedNumber) < 3:
```

Cron : 개요

- 유닉스 계열 (Linux, Mac OS 등) 운영체제 컴퓨터에서 시간을 기반으로 한 **작업 예약 스케줄러**
- 주기적으로 고정된 시간, 날짜, 간격에 원하는 프로그램을 실행 할 수 있도록 도와줌
- 서버는 꺼지지 않고 항상 작동 중인 것을 활용하여, 주로 서버에 적용
 - 평소에는 실행되지 않고 대기하다가, 특정 시간이 되면 해당 프로그램을 실행

Cron : 형식

🔗 프로그램 실행을 시간 형식에 맞게 적어 예약

시간 형식

Minute (0 ~ 59) : 분

Hour (0 ~ 23) : 시간

Day (1 ~ 31) : 일

Month (1 ~ 12) : 월

Week (0 ~ 6) : 요일 , 0은 일요일

예시

0 12 1 1 3

1월 1일 수요일 12시
정각에 실행

Cron : 활용

- ☛ 일정 간격의 모든 시간을 지정할 경우 : *

30 * * * * : 매시간 30분마다 실행

0 0 * * * : 매일 0시 0분마다 실행

- ☛ 일정 시간 간격을 반복하도록 지정할 경우 : ,

10,20,30 * * * * : 매시간 10분, 20분, 30분에 실행

Cron : 활용

🔧 시간 범위를 지정할 경우 : -

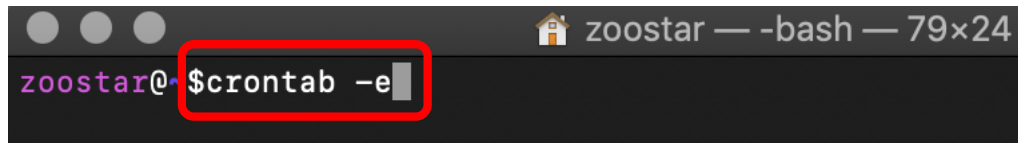
30 13-15 * * * : 13시,14시,15시 30분에 실행

🔧 시간 간격을 지정할 경우: /

0-30/5 * * * * : 매시간 0분부터 30분까지 5분 간격으로 실행

Cron 사용법 : UNIX 계열 운영체제

- 유닉스 계열(Linux, Mac OS 등) 운영체제에서는 별도의 설치없이 기본으로 사용 가능
- 터미널 창(콘솔)에서 **crontab -e** 명령어로 cron 설정 파일 편집
 - 현재 로그인 한 사용자에서 실행



```
zoostar ~ — -bash — 79x24
zoostar@ ~ % crontab -e
```

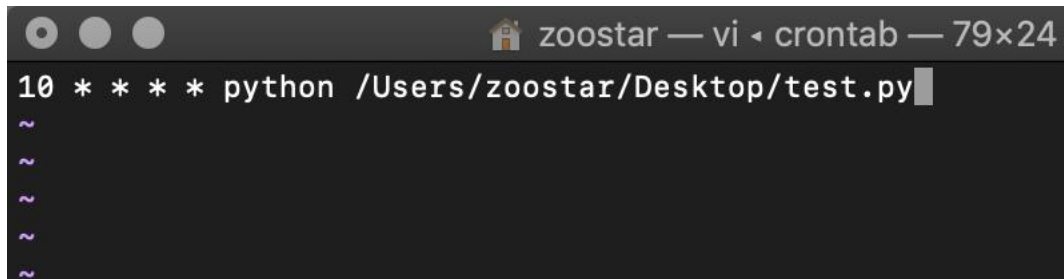


crontab -l : 예약된 작업의 목록 보기
crontab -r : 예약된 작업 삭제

Cron 사용법 : UNIX 계열 운영체제

✿ 작업 스케줄 설정 + 실행할 명령어

- 매 10분마다 바탕화면의 text.py 파이썬 파일 실행

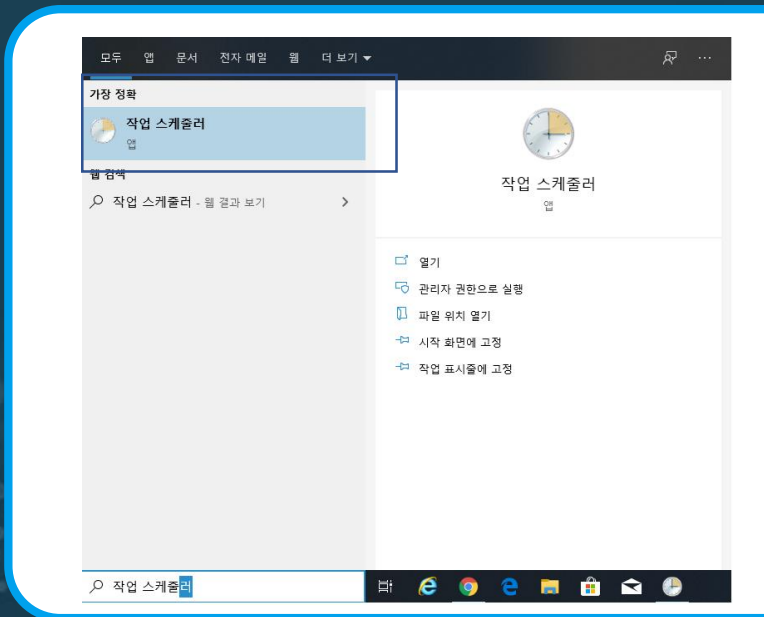
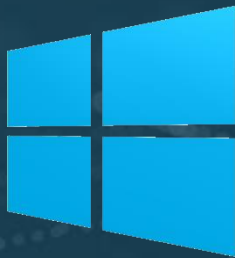


The screenshot shows a terminal window titled "zoostar — vi — crontab — 79x24". The terminal content displays a cron job entry: "10 * * * * python /Users/zoostar/Desktop/test.py". Below this entry, there are four tilde (~) characters, likely representing the rest of the crontab file.

```
zoostar — vi — crontab — 79x24
10 * * * * python /Users/zoostar/Desktop/test.py
~
~
~
~
```

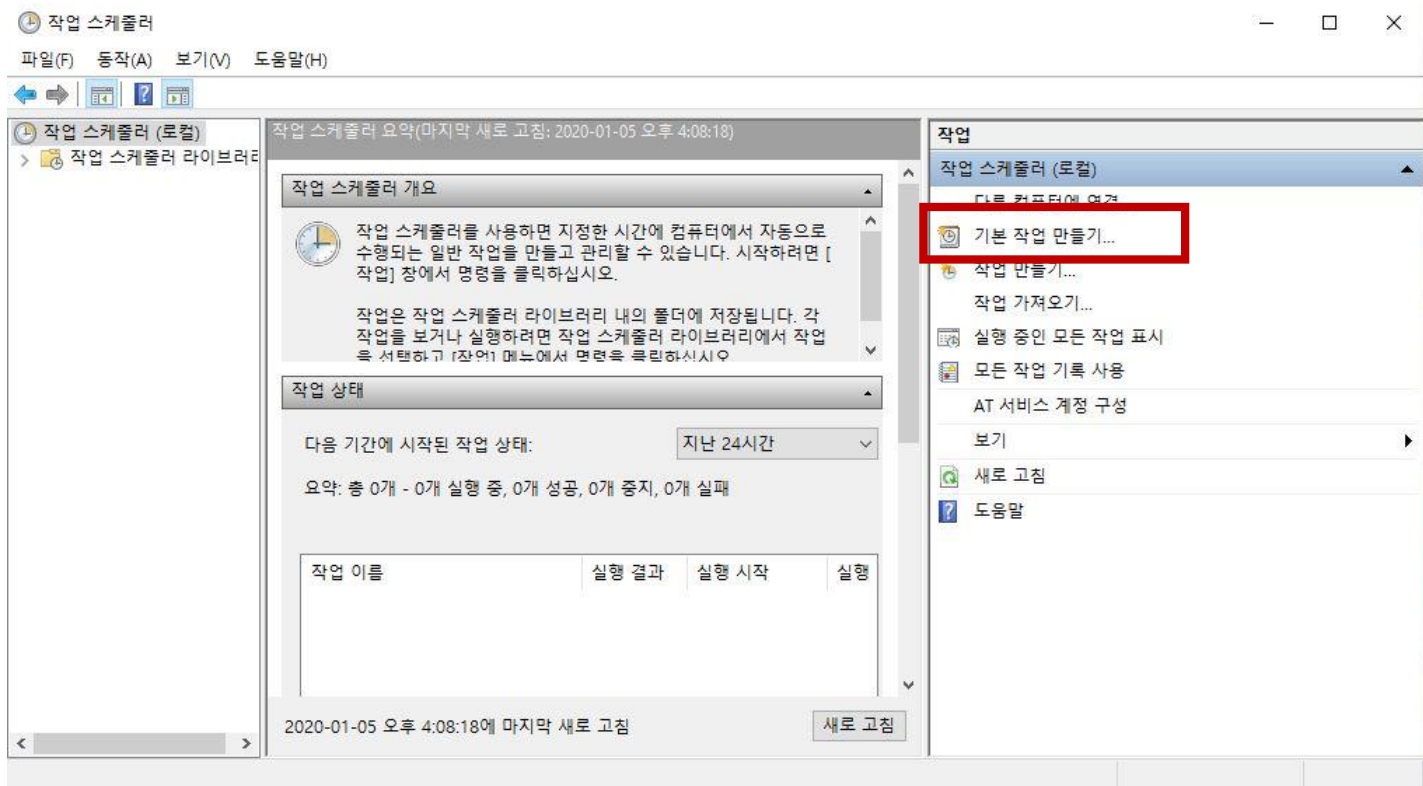
Cron 사용법 : Windows 계열 운영체제

- ❖ 별도의 Cron 프로그램 설치 필요
 - nncron, wincron ...
- ❖ Windows에서 Cron과 같은 역할을 하는 프로그램을 활용
 - 작업 스케줄러
- ❖ 명령어가 아닌 직관적인 GUI 환경으로 사용성이 좋음



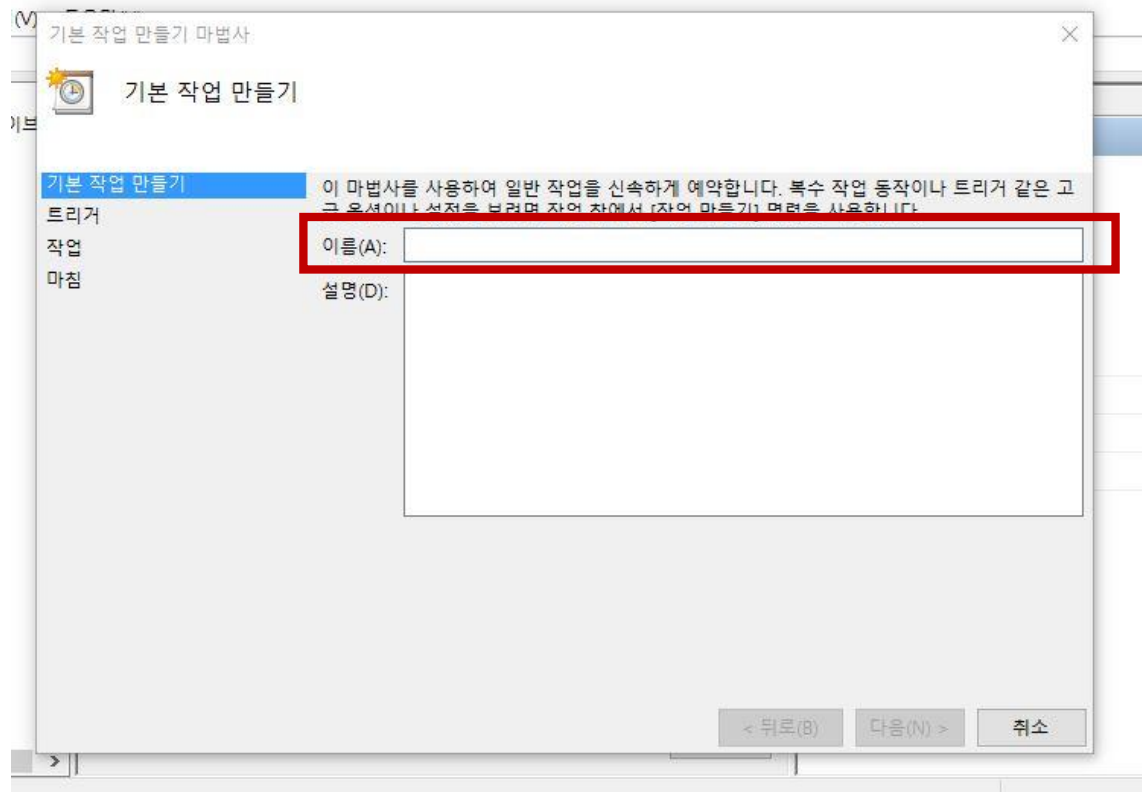
스케줄링 구현하기(Windows) Step1

작업 스케줄러 실행



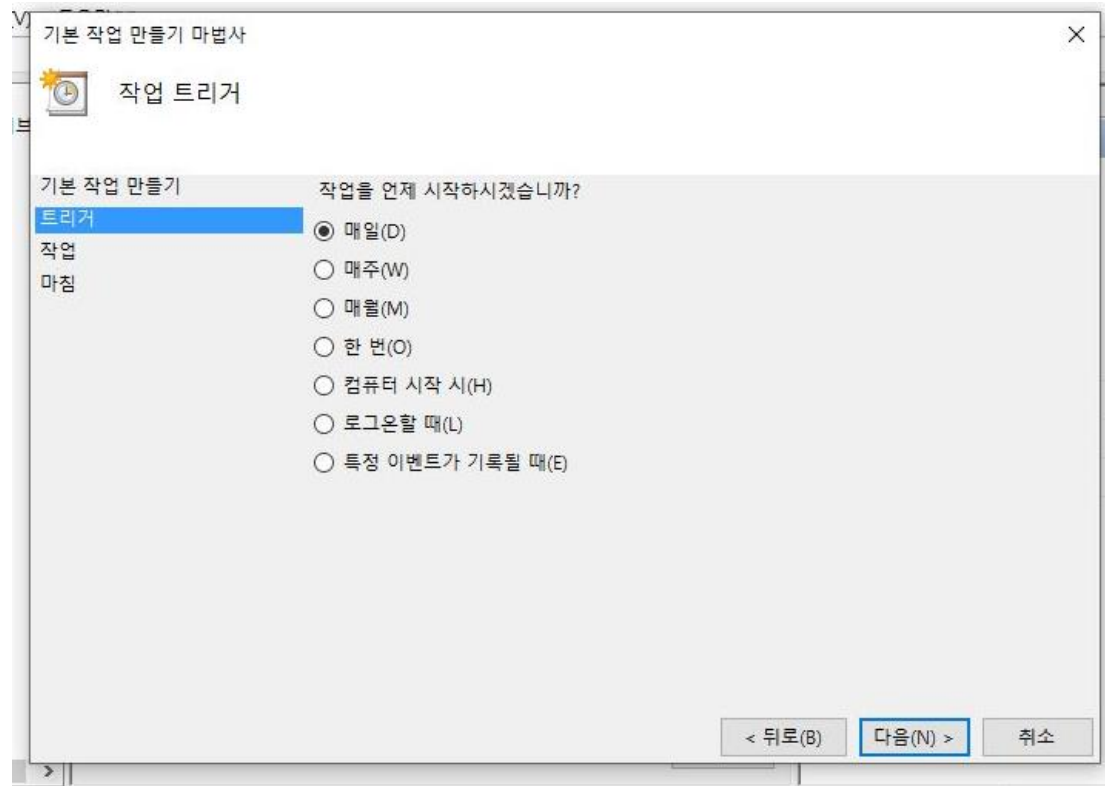
스케줄링 구현하기(Windows) Step1

기본 작업 만들기



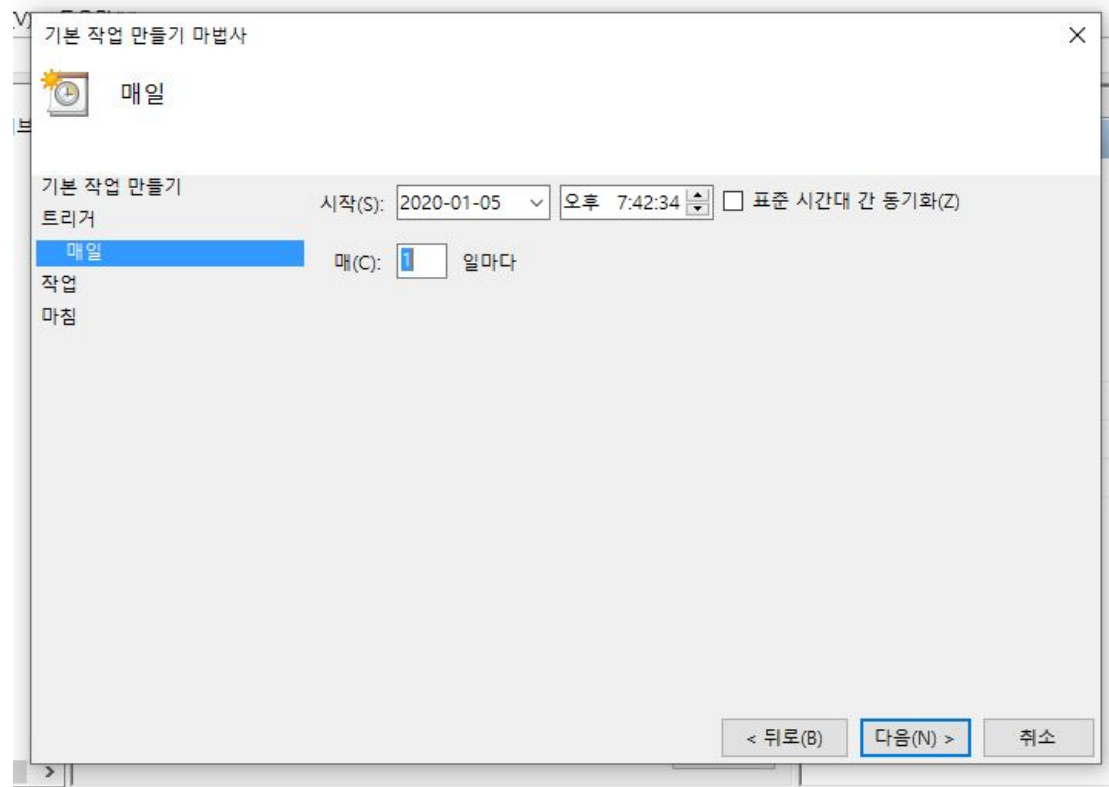
스케줄링 구현하기(Windows) Step2

트리거(실행 조건) 설정



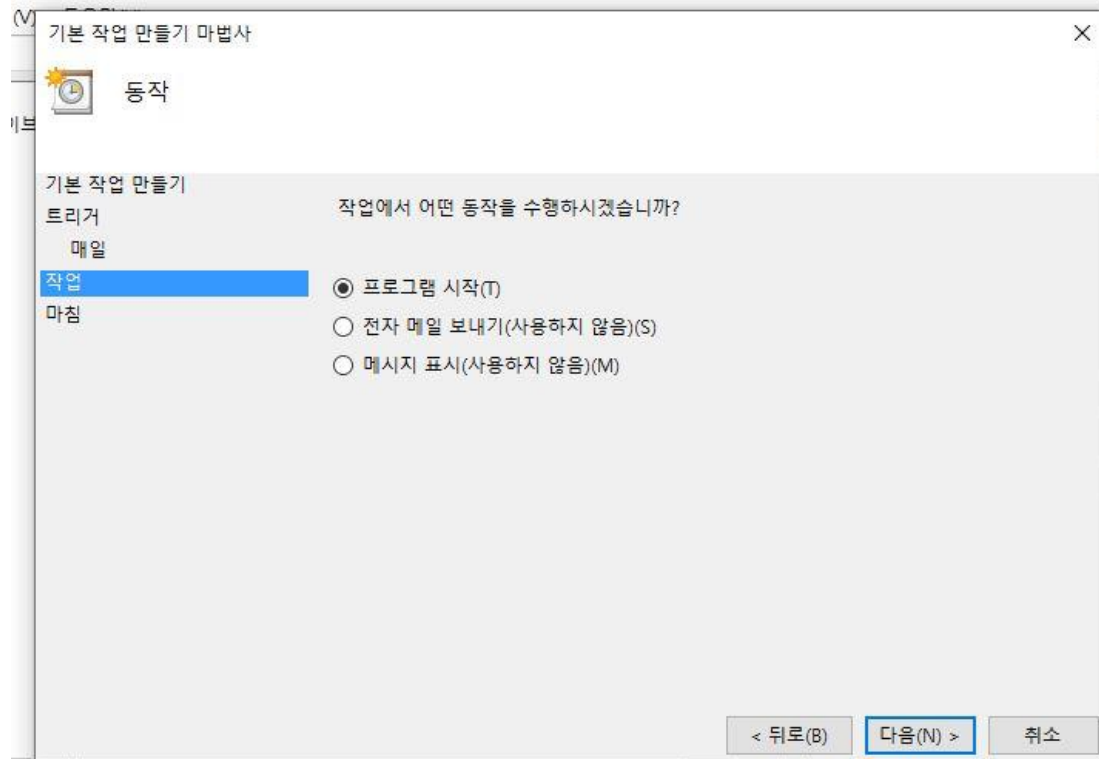
스케줄링 구현하기(Windows) Step2

트리거(실행 조건) 설정



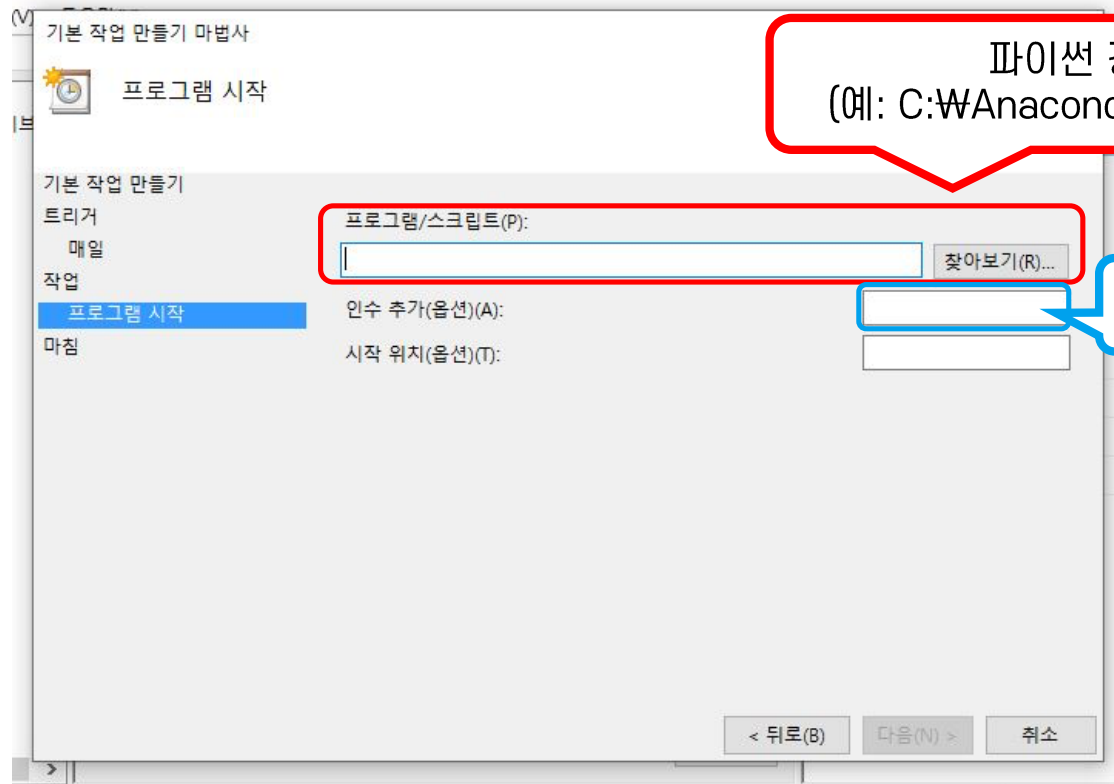
스케줄링 구현하기(Windows) Step3

실행할 작업 선택



스케줄링 구현하기(Windows) Step3

실행 파일 설정



LESSON

파이썬 모듈을 활용한 스케줄링

```
animLength = toTime - fromTime

# Ask user for directory
filePath = c4d.storage.SaveDialog()
filePath, objName = os.path.split(filePath)
objName = objName + "-"
filePath = filePath + "\\ "

# Ask for confirmation
questionDialogText = "Obj Sequence will be saved as:\n\n" \
    "" + filePath + objName + "####.obj\n\n" \
    "From frame " + str(fromTime) + " to " + str(toTime) + " "
proceedBool = c4d.gui.QuestionDialog(questionDialogText)

if proceedBool == True:

    # Loop through animation and export frames
    for x in range(0, animLength):

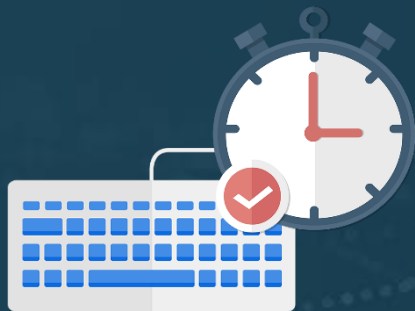
        # change frame, redraw view
        moveTime = c4d.BaseTime(fromTime, docFps) + c4d.BaseTime(x)
        doc.SetTime(moveTime)
        c4d.EventAdd(c4d.EVENT_FORCEREDRAW)
        c4d.DrawViews(c4d.DRAWFLAGS_FORCEFULLREDRAW)

        # update status bar
        c4d.StatusSetText("Exporting " + str(x) + " of " + str(animLength))
        c4d.StatusSetBar(100.0*x/animLength)

        # add buffer 0001
        bufferedNumber = str(doc.GetTime().GetFrame(docFps))
        if len(bufferedNumber) < 4:
```

Schedule 모듈 : 개요

- ❖ 파이썬에서 작업 예약을 하기 위한 모듈로
실행할 코드를 함수로 만들어서 호출하는 방식
- ❖ `pip install schedule` 명령어로 설치
- ❖ 명령어가 직관적으로 간단한 스케줄을 적용하기에 적합
 - 복잡한 스케줄은 작성하기 힘들



Schedule 모듈 : 활용

✦ 다음 세 가지를 조합하여 활용

- `every()`
: 매번 반복

- `seconds` : 초
- `minutes` : 분
- `hour` : 시
- `day` : 일
- `Monday - Sunday` : 요일

- `at()`
: 특정 시간

- 예시 : `스케줄.every().day.at(13:00)` : 매일 13시에 실행

✦ 스케줄 작성 후 `run_pending()` 함수로 시작

Schedule 모듈 : 활용

♣ schedule 구현 예시

무한 루프로
계속 실행 중

In [*]:

```
import schedule
import time

def job():
    print(time.time())

schedule.every(10).seconds.do(job)

while True:
    schedule.run_pending()
```

```
1578217284.858464
1578217294.85855
1578217304.858618
```

매 10초마다
함수 실행

Schedule 모듈 : 활용

- ♣ while 반복문을 활용하여 꾸준히 스케줄을 유지
- ♣ 특정 조건에 만족하는 스케줄이 있을 때마다 해당 함수 실행
- ♣ 해당 파이썬 파일을 계속 실행 시켜 두어야 한다는 단점이 있음

while 반복문을 이용한 스케줄링 예시

```
import schedule

def job():
    print("I'm working...")

#20분 마다 실행
schedule.every(20).minutes.do(job)

#매 시간 실행
schedule.every().hour.do(job)

#매일 10시 30분에 실행
schedule.every().day.at("10:30").do(job)

#매주 월요일 실행
schedule.every().monday.do(job)

#매주 수요일 13시 15분에 실행
schedule.every().wednesday.at("13:15").do(job)

while True:
    schedule.run_pending()
```

APScheduler 모듈 : 개요

- ♣ Advanced Python Scheduler 의 약자로 파이썬 코드를 주기적으로 수행 할 수 있도록 도와주는 외부 모듈
- ♣ MongoDB, Redis 등의 백엔드와 함께 사용할 수 있음(기본은 Memory)
 - 작업을 데이터베이스에 저장한 뒤 불러와서 사용 가능
- ♣ asyncio, gevent 등 여러가지 프레임워크와 연동을 지원
- ♣ pip install apscheduler 명령어로 설치 가능

APScheduler 모듈 : 개요

- ✦ scheduler 모듈과는 다르게 작업의 기간, 일시 정지, 다시 시작, 삭제 등 여러 가지 조건에 따라 스케줄링 가능
- ✦ 총 3가지 수행 방식을 지원

Cron 방식

Cron 표현식으로 코드를 수행

Date 방식

특정 날짜에 코드를 수행

Interval 방식

일정 주기로 코드를 수행

APScheduler 모듈 : 개요

스케줄러의 종류

- ① BlockingScheduler : 단일 작업 수행 시 사용
- ② BackgroundScheduler : 다수 작업 수행 시 사용
- ③ AsyncIOScheduler, GeventScheduler ... : 각 프레임워크 내 작업 수행 시 사용

스케줄러 객체를 생성 → add_job() 함수를 추가해 작업 예약 → start() 함수로 스케줄러 시작

```
from apscheduler.schedulers.background import BackgroundScheduler

sched = BackgroundScheduler()

def job():
    print("I'm working...")

sched.add_job(job, 'interval', seconds=3, id="1")

sched.start()

I'm working...
I'm working...
I'm working...
```

APScheduler 모듈 : 수행 방식별 사용법

🔌 **interval 방식** : start_date, end_date(해당 작업의 실행 시작~끝 지정)

```
from apscheduler.schedulers.blocking import BlockingScheduler

def job_function():
    print("Hello World")

sched = BlockingScheduler()

#매 2시간 간격으로 실행
sched.add_job(job_function, 'interval', hours=2)

sched.start()
```

Parameters

- **weeks** (*int*) – number of weeks to wait
- **days** (*int*) – number of days to wait
- **hours** (*int*) – number of hours to wait
- **minutes** (*int*) – number of minutes to wait
- **seconds** (*int*) – number of seconds to wait
- **start_date** (*datetime|str*) – starting point for the interval calculation
- **end_date** (*datetime|str*) – latest possible date/time to trigger on
- **timezone** (*datetime.tzinfo|str*) – time zone to use for the date/time calculations
- **jitter** (*int|None*) – advance or delay the job execution by **jitter** seconds at most.

APScheduler 모듈 : 수행 방식별 사용법

🔗 Cron 방식 : 문자열로도 표현 가능

```
from apscheduler.schedulers.blocking import BlockingScheduler

def job_function():
    print "Hello World"

sched = BlockingScheduler()

#6월부터 8월까지, 11월부터 12월까지 매 셋째 주 금요일, 0시부터 3시까지 한시간 간격으로 실행
sched.add_job(job_function, 'cron', month='6-8,11-12', day='3rd fri', hour='0-3')

sched.start()
```

Parameters

- `year (int|str)` - 4-digit year
- `month (int|str)` - month (1-12)
- `day (int|str)` - day of the (1-31)
- `week (int|str)` - ISO week (1-53)
- `day_of_week (int|str)` - number or name of weekday (0-6 or mon,tue,wed,thu,fri,sat,sun)
- `hour (int|str)` - hour (0-23)
- `minute (int|str)` - minute (0-59)
- `second (int|str)` - second (0-59)
- `start_date (datetime|str)` - earliest possible date/time to trigger on (inclusive)
- `end_date (datetime|str)` - latest possible date/time to trigger on (inclusive)
- `timezone (datetime.tzinfo|str)` - time zone to use for the date/time calculations (defaults to scheduler timezone)
- `jitter (int|None)` - advance or delay the job execution by `jitter` seconds at most.

APScheduler 모듈 : 수행 방식별 사용법

🔗 **date 방식** : 특정 시간에 작업 수행, args로 매개변수 전달

```
from apscheduler.schedulers.blocking import BlockingScheduler

sched = BlockingScheduler()

def my_job(text):
    print(text)

# 2020년 12월 31일 정각에 text 문구 출력
sched.add_job(my_job, 'date', run_date=date(2020, 12, 31), args=['text'])

sched.start()
```

Parameters

- **run_date** (*datetime|str*) – the date/time to run the job at
- **timezone** (*datetime.tzinfo|str*) – time zone for **run_date** if it doesn't have one already