포트폴리오

2025.09



# 데이터 파싱을 이용한 전국 산 정보 조회

**Presenter:** 박장호 | **Phone:** 010-4825-9941 | **E-mail:** 35379289p@gmail.com

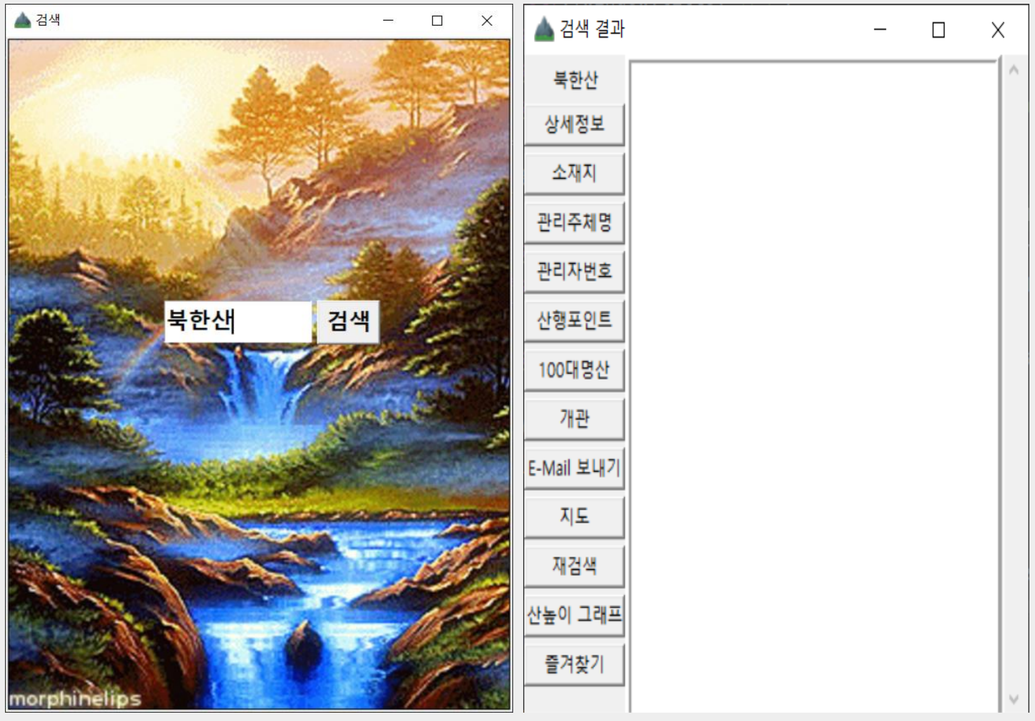
포트폴리오

• index

# 목차

포트폴리오

# 01 프로그램 개요




**개발 기간 : 2019.5.22~2019.6.12**

**개발 인원 : 2인**

**사용 기술 : Python 및 각종 모듈(tkinter 등)**

**Github : https://github.com/jangho-park-dev/ScriptLanguageTermProject**

**Youtube : https://youtu.be/YSPGoY6bQIs**

- 산 이름을 검색창에 입력 후 검색하면 공공 데이터 포털 API를 이용해 데이터를 파싱합니다.
- 파싱한 데이터를 통해 산의 상세 정보나, 정보를 이용한 지도·그래프·이메일 등의 기능을 사용할 수 있습니다.

# 02 메인 화면


```python
def InitTitle(self):        # 타이틀 윈도우
    self.Twindow = Tk()
    self.Twindow.iconbitmap(default='icon.ico')

    self.f1 = PhotoImage(file="anime01.png")
    self.f2 = PhotoImage(file="anime02.png")
    self.f3 = PhotoImage(file="anime03.png")
    self.f4 = PhotoImage(file="anime04.png")
    self.f5 = PhotoImage(file="anime05.png")
    self.f6 = PhotoImage(file="anime06.png")
    self.n = 0
    self.fn = self.f1

    self.Twindow.title("검색")
    self.Twindow.geometry("480x640+700+100")
    self.Tcanvas = Canvas(self.Twindow, width=480, height=640, relief="solid", bd=1)
    self.TempFont = font.Font(size=16, weight='bold', family='Consolas')

    Button(self.Twindow, text="검색",
           font=self.TempFont, command=self.nextWindow).place(x=295, y=250)
    self.e = Entry(self.Twindow, font=self.TempFont)
    self.e.place(x=150, y=250, width=140, height=40)

    self.anim_id = self.Twindow.after(0, self.Animation)

    self.Tcanvas.pack()
    self.Twindow.mainloop()
```

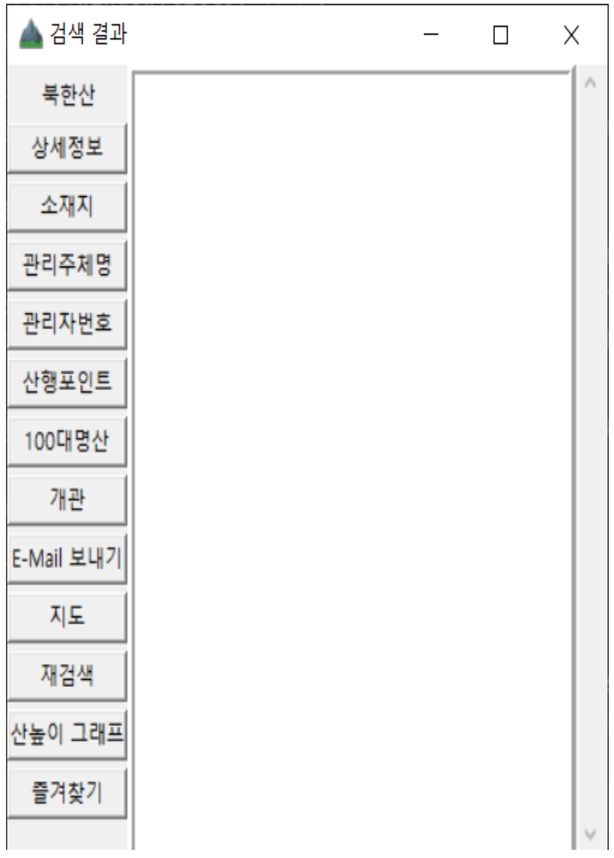
> “국내에 존재하는 산의 이름을 입력”

산의 이름을 입력하여 검색 버튼을 누르면 검색 결과 창으로 넘어갑니다.

메인 화면 애니메이션을 위한 이미지 사전 저장 및 마지막의 after 함수에 Animation 함수를 bind합니다.

검색 버튼을 누르면 nextWindow 함수가 실행되도록 bind합니다.

# 03 동작 화면



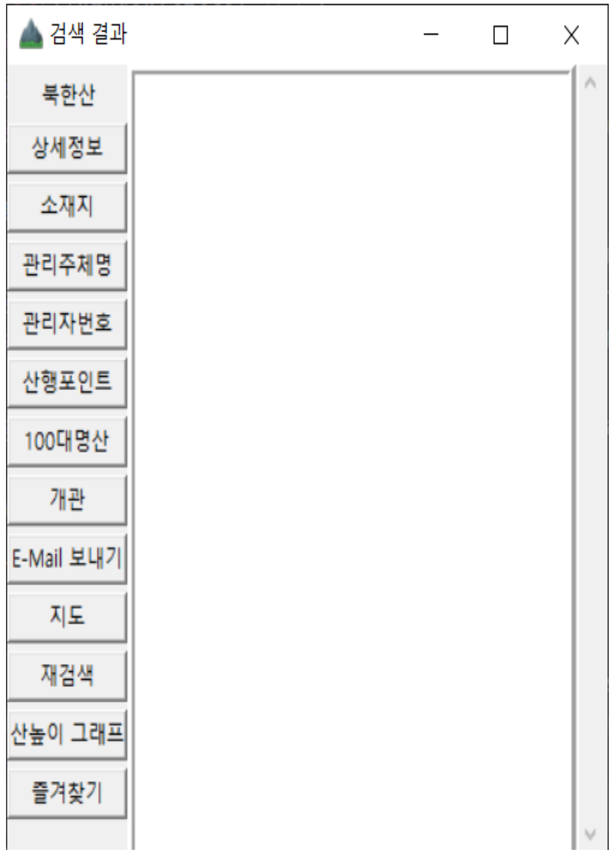

```python
def nextWindow(self):           # 검색 버튼 누르면 실행되는 함수
    self.MountainName = self.e.get()        # 타이틀에서 산 이름 받아옴
    self.search_word = urllib.parse.quote(self.e.get())
    ...
    conn = http.client.HTTPConnection("openapi.forest.go.kr")
    service_key = "cuVGydw6yzwC%2B6YdfYKOPzXxvC45arm%2F1M1dpN31ZrgomqlojiWkwCq0jZqneeAvo
    url = (
        "https://apis.data.go.kr/1400000/service/cultureInfoService2/mntInfoOpenAPI2"
        f"?serviceKey={service_key}&searchWrd={self.search_word}&pageNo=1&numOfRows=10"
    )

    conn.request("GET", url)
    req = conn.getresponse()
    self.tree = ElementTree.fromstring(req.read().decode('utf-8'))

    try:
        if hasattr(self, 'anim_id') and self.anim_id:
            self.Twindow.after_cancel(self.anim_id)
    except Exception:
        pass

    self.Twindow.destroy()    # 기존에 있던 타이틀 윈도우 파괴
    self.InitResult()         # 결과창 생성
```

**“ 검색한 산으로 데이터를 파싱 ”**

검색 버튼을 누르면 공공 데이터 포털의 URL 및 API key로 새 URL을 만들어 데이터를 파싱할 수 있도록 합니다.

메인 애니메이션 창이 닫힐 때, 저장해놨던 콜백 함수 id를 통해 after가 확실하게 종료될 수 있도록 합니다.

# 03 동작 화면

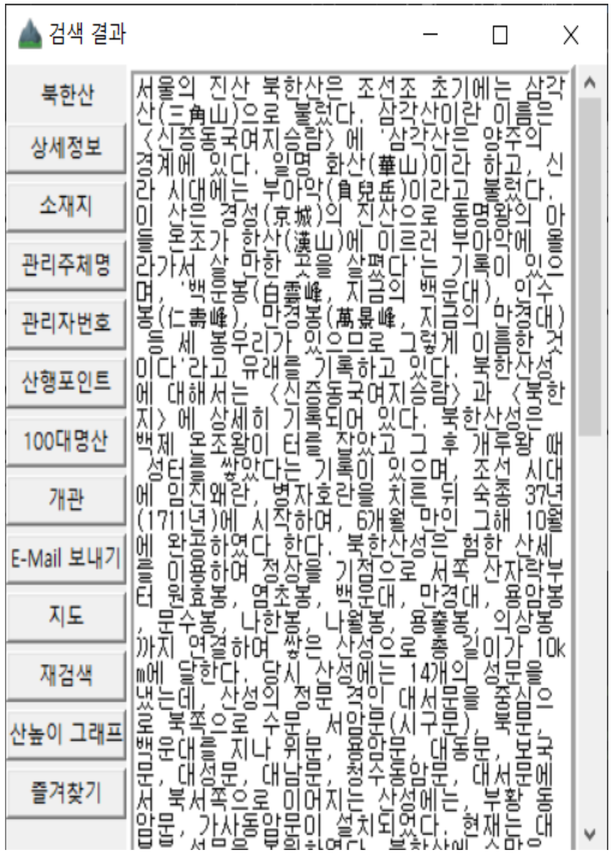



```
def InitResult(self):        # 결과창 생성
    self.window = Tk()
    self.window.iconbitmap(default='icon.ico')
    self.window.title("검색 결과")
    self.window.geometry("400x402+700+100")
    self.MapCanvas = Canvas(self.window, width=800, height=402)
    self.MapCanvas.pack()
    self.TempFont = font.Font(size=16, weight='bold', family='Consolas')

    Label(self.window, text=self.MountainName).place(x=20, y=5)

    Button(self.window, text="상세정보", width=10, command=self.Information).place(x=0, y=30)
    Button(self.window, text="소재지", width=10, command=self.Address).place(x=0, y=60)
    Button(self.window, text="관리주체명", width=10, command=self.MountainAdmin).place(x=0, y=90)
    Button(self.window, text="관리자번호", width=10, command=self.MountainAdminNum).place(x=0, y=1
    Button(self.window, text="산행포인트", width=10, command=self.HikingPoint).place(x=0, y=150)
    Button(self.window, text="100대명산", width=10, command=self.SpecialMountain).place(x=0, y=18
    Button(self.window, text="개관", width=10, command=self.Survey).place(x=0, y=210)
    Button(self.window, text="E-Mail 보내기", width=10, command=self.sendMail).place(x=0, y=240)
    Button(self.window, text="지도", width=10, command=self.Map).place(x=0, y=270)
    Button(self.window, text="재검색", width=10, command=self.reSearch).place(x=0, y=300)
    Button(self.window, text="산높이 그래프", width=10, command=self.Graph).place(x=0, y=330)
    Button(self.window, text="즐겨찾기", width=10, command=self.Favorite).place(x=0, y=360)

    scroll = Scrollbar(self.window)
    self.text = Text(self.window, width=41, height=32, borderwidth=5, relief="ridge", yscrollco
    scroll.place(x=380, y=0, height=402)
    self.text.place(x=80, y=0)
```

## “ UI 생성 ”

새 창이 열릴 때 필요한 캔버스 및 레이블, 버튼들을 생성하고, 각 버튼의 기능에 맞는 함수들을 bind합니다.

# 03 동작 화면

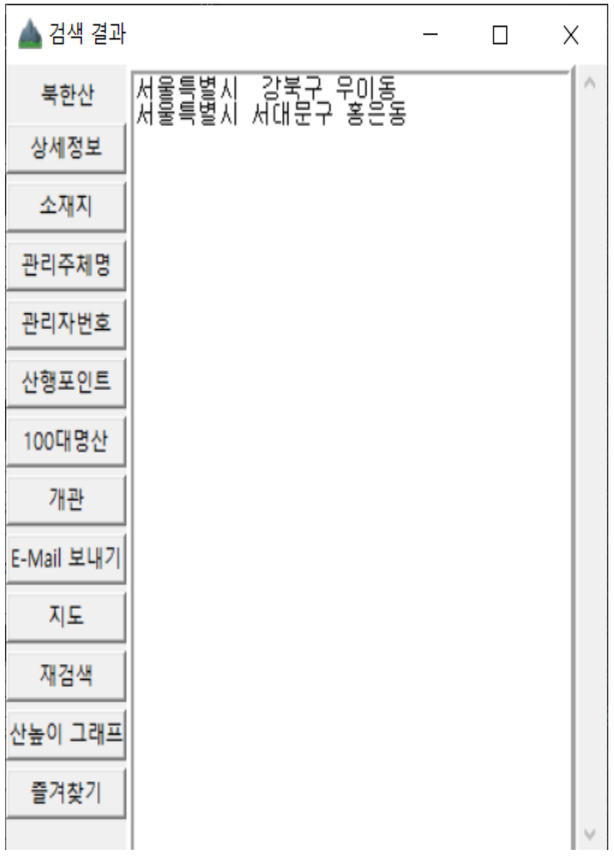



```
def Information(self):
    self.text.delete(1.0, END)

    items = list(self.tree.iter("item"))
    if not items:
        self.text.insert(1.0, "검색 결과가 없습니다.")
        return

    for item in items:
        # 여러 가능한 태그명(구버전/신버전 대비)
        raw_info = self._get_text(item, 'mntninfodtlinfoc', 'mntidetails', 'mntninfodscrt', 'mnt
        info = self._clean_text(raw_info)
        height = self._get_text(item, 'mntninfohght', 'mntihigh')
        sub_name = self._get_text(item, 'mntnsbttlinfo', 'mntisname')
        name = self._get_text(item, 'mntnnm', 'mntiname',)

        # 안전하게 삽입
        if name:
            self.text.insert(1.0, name + '\n\n')
        if sub_name:
            self.text.insert(1.0, "산의 부제 : " + self._clean_text(sub_name) + '\n\n')
        if height:
            self.text.insert(1.0, "높이 : " + height + '\n\n')
        if info:
            self.text.insert(1.0, info + '\n\n')

    global TEXT
    TEXT = self.text.get(1.0, END)
```

## 상세정보

파싱한 데이터에서 item별 선정을 통해 list를 만들고, 각 항목명에 대응하는 요소를 저장 후 text에 삽입합니다.

# 03 동작 화면

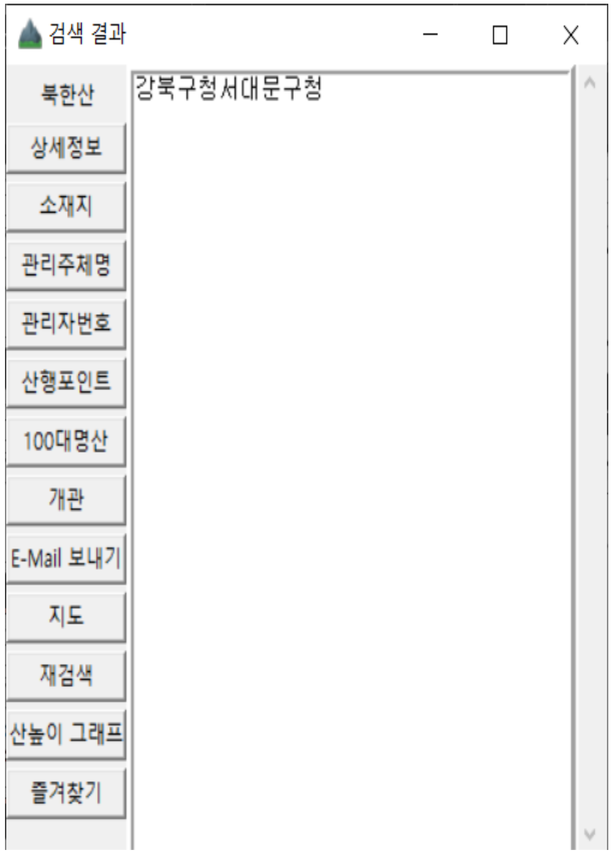



```
def Address(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.MountainAddress = item.find("mntiadd")
        self.text.insert(1.0, self.MountainAddress.text + '\n')
```

## “ 소재지 ”

파싱한 데이터에서 item을 가져와 소재지에 해당하는 요소를 text에 삽입합니다.

# 03 동작 화면



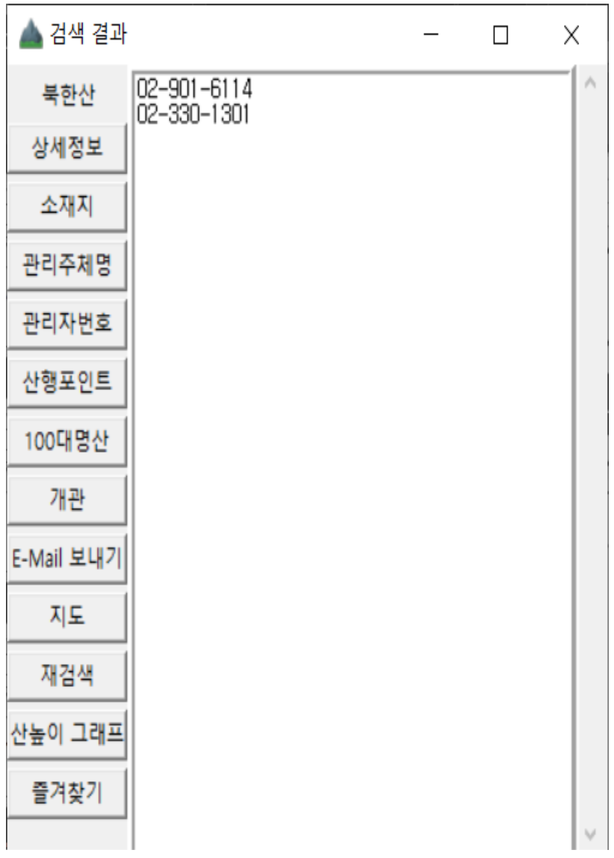

```
def MountainAdmin(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.AdminInfo = item.find("mntiadmin")
        self.L.append(self.AdminInfo.text)
        self.AdminInfo.text = self.AdminInfo.text.replace('<BR>', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('br /', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('&lt;', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('&gt;', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('&amp;', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('nbsp;', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('<p>&', '\n')
        self.AdminInfo.text = self.AdminInfo.text.replace('</p>', '\n')
        self.text.insert(1.0, self.AdminInfo.text)

    if not self.L or self.L[0] == '':
        self.text.insert(1.0, "관리자 정보가 없습니다.")

    self.L.clear()
```

## “관리주체명”

파싱한 데이터에서 item을 가져와 관리주체명에 해당하는 요소를 text에 삽입합니다.

본 코드 및 이하 코드에서 item 내부 문자열에 있는 문자를 개행 문자로 대체하는 부분이 있을 수 있습니다.

# 03 동작 화면

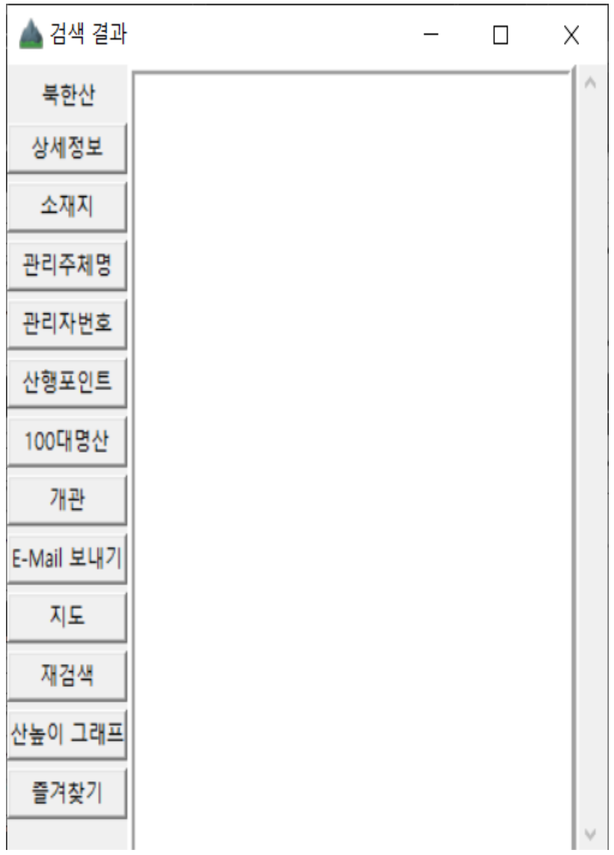



```
def MountainAdminNum(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.AdminNumInfo = item.find("mntiadminnum")
        self.L.append(self.AdminNumInfo.text)
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('<BR>', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('br /', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('&lt;', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('&gt;', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('&amp;', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('nbsp;', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('<p>&', '\n')
        self.AdminNumInfo.text = self.AdminNumInfo.text.replace('</p>', '\n')
        self.text.insert(1.0, self.AdminNumInfo.text + '\n')

    if not self.L or self.L[0] == '':
        self.text.insert(1.0, "관리자 전화번호가 없습니다.")

    self.L.clear()
```

**“ 관리자 전화번호 ”**

파싱한 데이터에서 item을 가져와 관리자 전화번호에 해당하는 요소를 text에 삽입합니다.

# 03 동작 화면



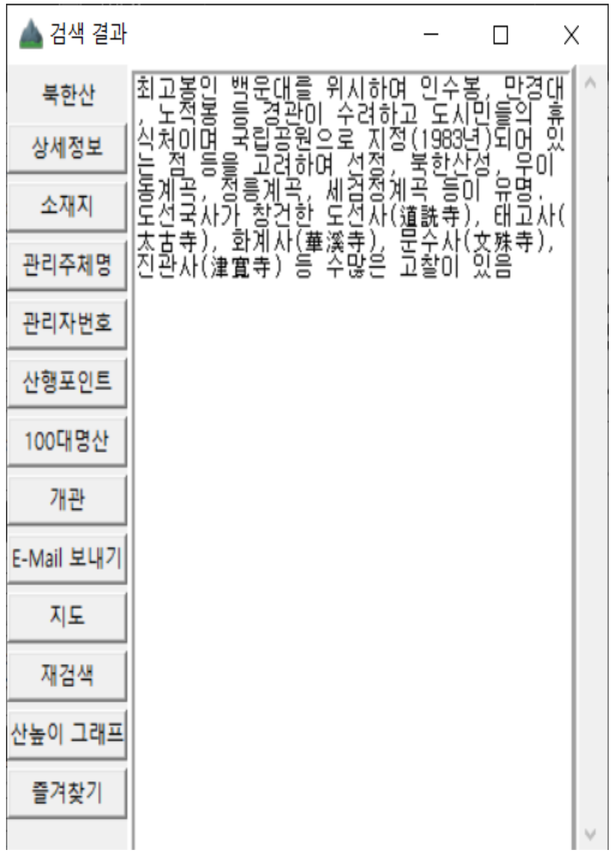

```
def HikingPoint(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.HikingPointInfo = item.find("frtrlsectnnm")
        self.L.append(self.HikingPointInfo.text)
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('<BR>', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('br /', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('&lt;', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('&gt;', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('&amp;', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('nbsp;', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('<p>&', '\n')
        self.HikingPointInfo.text = self.HikingPointInfo.text.replace('</p>', '\n')
        self.text.insert(1.0, self.HikingPointInfo.text)

    if not self.L or self.L[0] == '':
        self.text.insert(1.0, "산행 포인트 정보가 없습니다.")

    self.L.clear()
```

**“ 산행포인트 ”**

파싱한 데이터에서 item을 가져와 산행 포인트에 해당하는 요소를 text에 삽입합니다.

파싱한 데이터에 따라, 데이터가 있는데 해당 데이터가 공백인 경우 결과창이 공백으로 보일 수 있습니다.

# 03 동작 화면

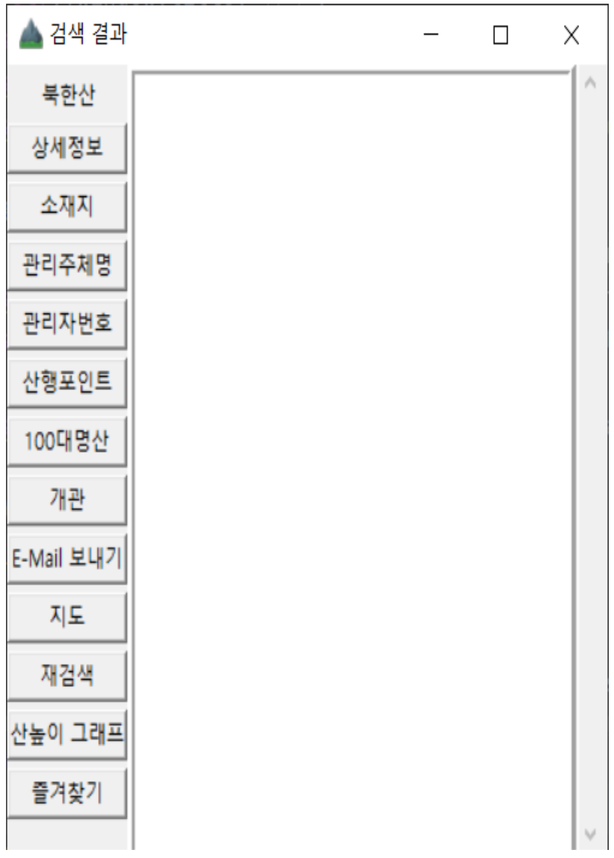



```
def SpecialMountain(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.SM = item.find("mntitop")
        self.SM.text = self.SM.text.replace('<BR>', '\n')
        self.SM.text = self.SM.text.replace('br /', '\n')
        self.SM.text = self.SM.text.replace('&lt;', '\n')
        self.SM.text = self.SM.text.replace('&gt;', '\n')
        self.SM.text = self.SM.text.replace('&amp;', '\n')
        self.SM.text = self.SM.text.replace('nbsp;', '\n')
        self.text.insert(1.0, self.SM.text)
```

“ 100대 명산 ”

파싱한 데이터에서 item을 가져와 100대 명산에 해당하는 요소를 text에 삽입합니다.

# 03 동작 화면

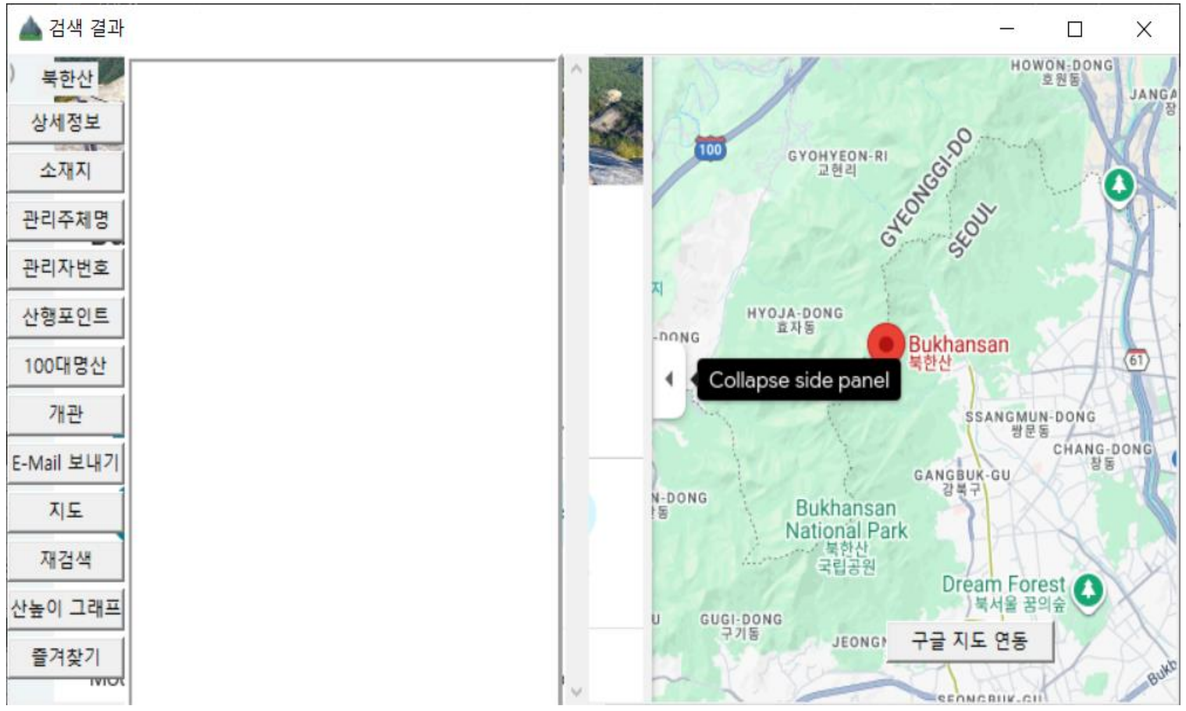



```
def Survey(self):
    self.text.delete(1.0, 1000.0)
    for item in self.tree.iter("item"):
        self.Survey = item.find("mntisummary")
        self.Survey.text = self.Survey.text.replace('<BR>', '\n')
        self.Survey.text = self.Survey.text.replace('br /', '\n')
        self.Survey.text = self.Survey.text.replace('&lt;', '\n')
        self.Survey.text = self.Survey.text.replace('&gt;', '\n')
        self.Survey.text = self.Survey.text.replace('&amp;', '\n')
        self.Survey.text = self.Survey.text.replace('nbsp;', '\n')

        self.text.insert(1.0, self.Survey.text)

        if not self.L or self.L[0] == '':
            self.text.insert(1.0, "개관 정보가 없습니다.")
```

## “ 개관 ”

파싱한 데이터에서 item을 가져와 개관에 해당하는 요소를 text에 삽입합니다.
파싱한 데이터에 따라, 데이터가 있는데 해당 데이터가 공백인 경우 결과창이 공백으로 보일 수 있습니다.

# 03 동작 화면




## 지도

지도 버튼을 누르면 구글 Geocoding API 및 pdfcrowd API와 연동되어, 해당 산이 검색된 구글 맵의 스크린샷을 이미지 파일로 저장함과 동시에, 그 이미지의 일부를 어플리케이션에 보여줍니다.

# 03 동작 화면

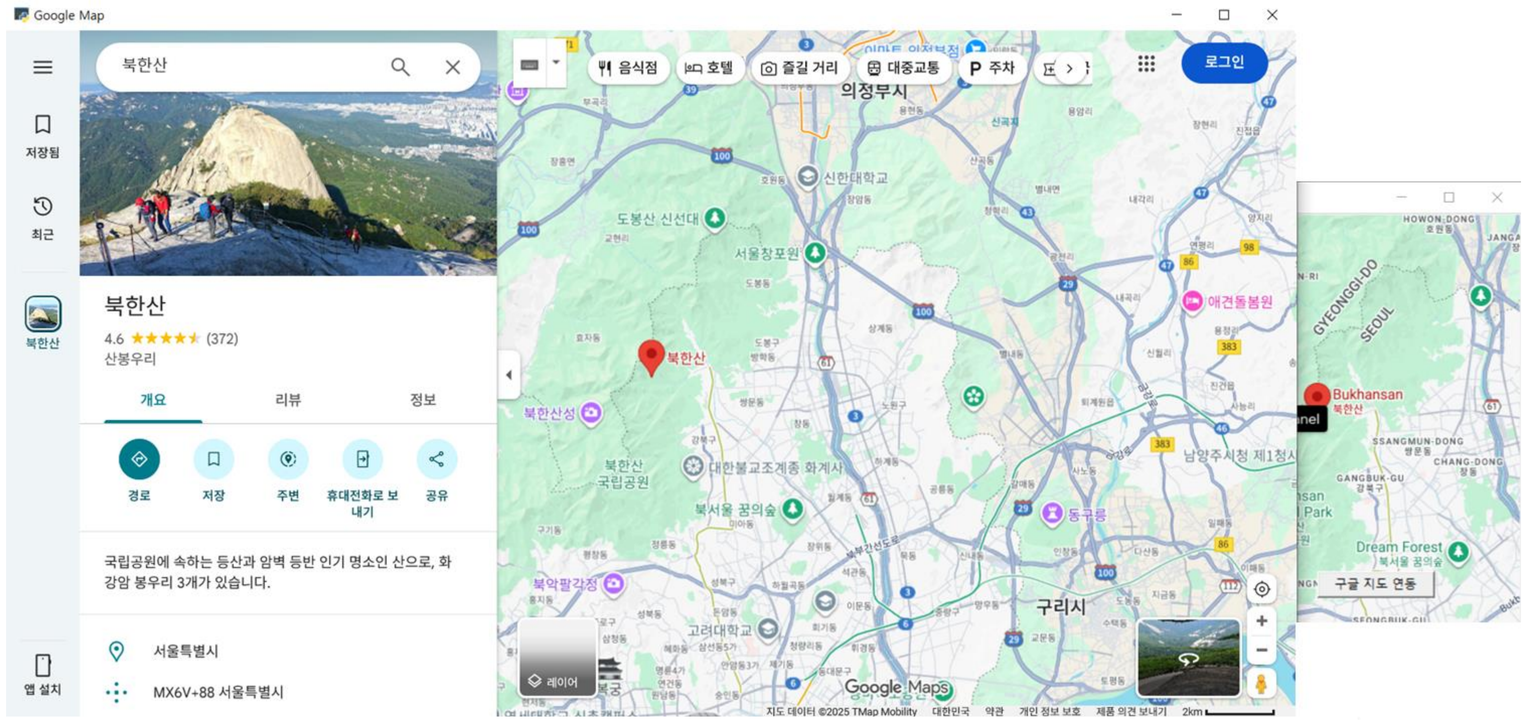
```
def Map(self):
    import pdfcrowd
    import sys

    self.URL = 'https://maps.googleapis.com/maps/api/geocode/json?key=' \
               'AIzaSyBFVqFYHLQNOYuSVfkiHCv1GkyfUpnpAIY&sensor=false&language=ko&address={}'\
        .format(self.MountainName)

    self.response = requests.get(self.URL)

    self.data = self.response.json()

    self.lat = self.data['results'][0]['geometry']['location']['lat']
    self.lng = self.data['results'][0]['geometry']['location']['lng']

    self.map_url = 'https://www.google.co.kr/maps/search/' + self.MountainName + '/@' + \
                   str(self.lat) + ',' + str(self.lng) + ',12z'

    messagebox.showinfo("알림", "지도 그리는 중...")

    # html file to png file
    try:
        # Create an API client instance.
        client = pdfcrowd.HtmlToImageClient('janghoparkdev', 'd63ba72280fa2a4edce5a6813af47934')

        # Configure the conversion.
        client.setOutputFormat('png')
        client.setScreenshotWidth(1280)
        client.setScreenshotHeight(720)

        # Run the conversion and save the result to a file.
        client.convertUrlToFile(self.map_url, 'Searched_Result_Map.png')
```

```
    except pdfcrowd.Error as why:
        sys.stderr.write('PDFCrowd Error: {}\n'.format(why))
        raise

    self.image = PhotoImage(file='Searched_Result_Map.png')
    self.window.geometry("800x402")
    self.MapCanvas.create_image(600, 201, image=self.image)

    messagebox.showinfo("알림", "지도가 완성되었습니다!")

    Button(self.window, text="구글 지도 연동", overrelief="solid", width=15,
           command=self.WebViewer).place(x=600, y=350)
```

## “ 지도 ”

Geocode API로 데이터를 파싱하여 검색한 산의 위도와 경도를 저장합니다.
저장한 값으로 URL을 새로 만들고, pdfcrowd API로 해당 URL 사이트를 png파일로 만들어 저장합니다.
버튼에 bind되어있는 WebViewer함수, 즉 구글 지도 연동은 후술하겠습니다.

# 03 동작 화면




## 지도(구글 지도 연동)

어플리케이션에 나타났던 지도의 아래에 구글 지도 연동 버튼이 생성됩니다.
해당 버튼을 누르면 웹뷰어 윈도우가 새 창으로 열리며 구글 맵에 해당 산을 검색해 보여줍니다.

# 03 동작 화면

```
def WebViewer(self):
    subprocess.Popen([sys.executable, 'webview_launcher.py', self.map_url])
```

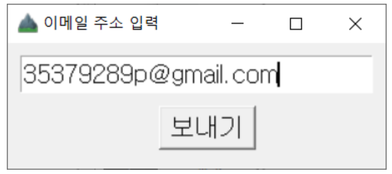
webview_launcher.py

```
import sys
import webview

if len(sys.argv) < 2:
    print("URL이 필요합니다.")
    sys.exit(1)

url = sys.argv[1]

webview.create_window('Google Map', url, width=1280, height=720)
webview.start()
```

“ 지도(구글 지도 연동) ”

구글 지도 연동 버튼을 누르면 WebViewer 함수가 실행되고, webview_launcher.py는 Web View용 윈도우를 생성합니다.

# 03 동작 화면




```
def sendMail(self):
    # global value
    global MAIL
    self.Twindow2 = Tk()
    self.Twindow2.title("이메일 주소 입력")
    self.Twindow2.geometry("300x100+700+250")
    self.TempFont2 = font.Font(size=5, weight='bold', family='Consolas')

    Button(self.Twindow2, text="보내기",
           font=self.TempFont2, command=self.mailSend).place(x=120, y=50)

    self.e2 = Entry(self.Twindow2, font=self.TempFont2)
    self.e2.place(x=10, y=10, width=280, height=30)
```

**“ E-Mail 보내기 ”**

E-Mail 주소를 받기 위한 윈도우를 생성합니다.

보내기 버튼을 누르면 전송을 시작합니다.

# 03 동작 화면

산 상세정보 받은편지함 ×

35379289p@gmail.com
나에게

서울의 진산 북한산은 조선조 초기에는 삼각산(三角山)으로 불렀다. 삼각산이란 이름은
불렀다. 이 산은 경성(京城)의 진산으로 동명왕의 아들 온조가 한산(漢山)에 이르러 부아
금의 만경대) 등 세 봉우리가 있으므로 그렇게 이름한 것이다'라고 유래를 기록하고 있다.
고 그 후 개루왕 때 성터를 쌓았다는 기록이 있으며, 조선 시대에 임진왜란, 병자호란을 치
을 기점으로 서쪽 산자락부터 원효봉, 염초봉, 백운대, 만경대, 용암봉, 문수봉, 나한봉, 나
의 정문 격인 대서문을 중심으로 북쪽으로 수문, 서암문(시구문), 북문, 백운대를 지나 위
문이 설치되었다. 현재는 대부분 성문을 복원하였다. 북한산에 수많은 등산로 거미줄 같이
4.19탑 기점, 정릉 기점, 세검정 기점, 불광동 기점, 구파발 북한산성 기점으로 구분할 수
조사대상지역인 서울 서대문구 행정구역의 지형은 서울 서대문구 홍은2동지역을 감싸고
구와 은평구로 행정구역이 나눠어지며, 구기터널 부근부터는 성벽을 따라 서대문구와 종
지역은 다소 지세가 험한 편이다. 일반등산로의 경우 그다지 등산이 어렵지는 않으나, 바
함될 정도로 자연경관이 매우 뛰어난 편이다. 비록 조사대상지역이 북한산끝자락에 속하

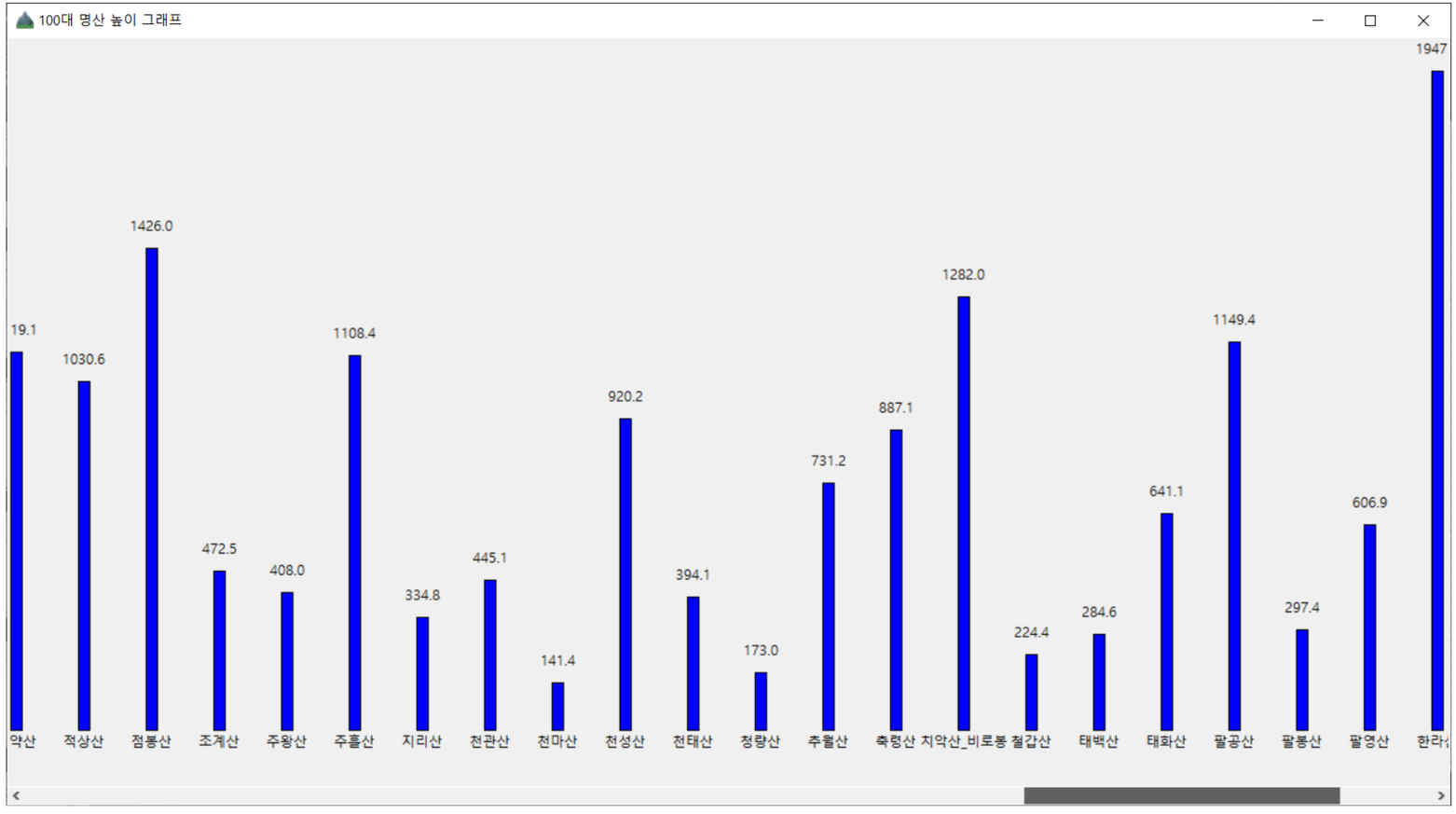
첨부파일 1개 • Gmail에서 스캔함

```python
def mailSend(self):
    global MAIL
    global TEXT
    MAIL = self.e2.get()
    host = "smtp.gmail.com"  # Gmail SMTP 서버 주소.
    port = "587"
    messagebox.showinfo("Loading", "이메일 보내는 중...")

    senderAddr = "35379289p@gmail.com"  # 보내는 사람 email 주소.
    recipientAddr = MAIL  # 받는 사람 email 주소.

    msg = MIMEMultipart()

    msg['Subject'] = "산 상세정보"
    msg['From'] = senderAddr
    msg['To'] = recipientAddr

    self.Information()      # 상세정보 누르지 않아도 여기서 다시 실행

    # ----- 본문 추가 -----
    # TEXT 변수를 html 본문으로
    body = MIMEText(TEXT, "html", "utf-8")
    msg.attach(body)

    # ----- 이미지 첨부 -----
    with open("Searched_Result_Map.png", "rb") as f:
        img = MIMEImage(f.read(), _subtype="png")
        img.add_header("Content-Disposition", "attachment", filename="Searched_Result_Map.png")
        msg.attach(img)

    # 메일 전송
    s = mysmtplib.MySMTP(host, port)
    s.ehlo()
    s.starttls()
    s.login("35379289p@gmail.com", "lyqt vnld amnz xzvg")  # 앱 비밀번호
    s.sendmail(senderAddr, [recipientAddr], msg.as_string())
    s.close()

    messagebox.showinfo("Complete", "이메일 보내기 완료!")
```

## “ E-Mail 보내기 ”

E-Mail 보내기 버튼을 누르면, Gmail SMTP 서버와 연동해 산의 상세 정보와 지도 버튼을 눌렀을 때 저장했던 이미지를 해당 메일 주소로 보냅니다.

# 03 동작 화면




"산 높이 그래프"

100대 명산들의 높이 데이터를 파싱해 새 윈도우에 높이에 따른 그래프를 그립니다.

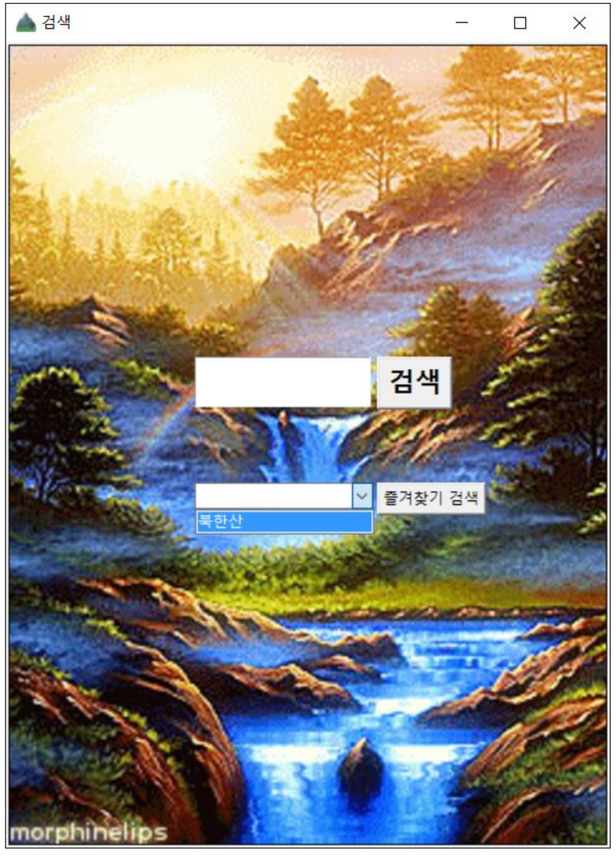
# 03 동작 화면

```python
def Graph(self):
    self.f = open("100대산(중복제외95).txt", 'r+')
    self.string = ''
    self.string += self.f.read()
    self.MountainList = []
    self.MountainList += self.string.split('\n')
    self.HeightList = []
    self.NameList = []

    conn = http.client.HTTPConnection("openapi.forest.go.kr")
    service_key = "cuVGydw6yzwC%2B6YdfYKOPzXxvC45arm%2F1M1dpN31ZrgomglojiWkwCq0jZqneeAvoEZxOqR8WrymypQQvq4hpg%3D%3
    graph_url = (
        "https://apis.data.go.kr/1400000/service/cultureInfoService2/mntInfoOpenAPI2"
        f"?serviceKey={service_key}&searchWrd="
    )
    url = ''
    messagebox.showinfo("알림", "확인을 누르면 그래프 그리기를 시작합니다.")

    for i in range(95):
        name = urllib.parse.quote(self.MountainList[i])
        url += graph_url + name + "&pageNo=1&numOfRows=10"
        conn.request("GET", url)
        req = conn.getresponse()
        graph_tree = ElementTree.fromstring(req.read().decode('utf-8'))
        url = ''
        for item in graph_tree.iter("item"):
            self.HeightList.append(item.find("mntihigh").text)
            self.NameList.append(item.find("mntiname").text)

    self.GraphWindow = Tk()
    self.GraphWindow.iconbitmap(default='icon.ico')
    self.GraphWindow.title("100대 명산 높이 그래프")
    self.GraphWindow.geometry("1280x680")
    self.GraphCanvas = Canvas(self.GraphWindow, width=1280, height=670)
    scrollbar = Scrollbar(self.GraphWindow, orient="horizontal", command=self.GraphCanvas.xview)
    self.GraphCanvas.configure(xscrollcommand=scrollbar.set)
    scrollbar.pack(side="bottom", fill="x")

    interval = 20
    for i in range(len(self.HeightList)):
        # 높이를 실수로 변환
        height = float(self.HeightList[i])
        # 캔버스는 정수 좌표만 받으니 int 변환
        rect_height = int(height * 0.3)
        self.GraphCanvas.create_rectangle(
            i * 10 + interval, 650,
            (i + 1) * 10 + interval, 650 - rect_height,
            outline="black", fill="blue"
        )
        self.GraphCanvas.create_text(i * 10 + interval + 5, 660, text=self.NameList[i])
        self.GraphCanvas.create_text(i * 10 + interval + 5, 650 - rect_height - 20, text=f"{height:.1f}")
        interval += 50

    self.f.close()
    self.NameList.clear()
    self.HeightList.clear()
    self.GraphCanvas.pack()
    self.GraphWindow.bind("<Configure>", self.on)
    self.GraphWindow.mainloop()

def on(self, event):
    self.GraphCanvas.configure(scrollregion=self.GraphCanvas.bbox("all"))
```

## “ 산 높이 그래프 ”

100대 명산(중복 제외 95)의 이름과 높이를 각각 파싱해 저장하고, 새 윈도우를 생성해 이름 기준 사전 순으로 정렬해 그래프를 그립니다.

# 03 동작 화면




```python
def Favorite(self):
    global favoriteList
    favoriteList.append(self.MountainName)
    messagebox.showinfo("알림", "즐겨찾기에 추가되었습니다.\n타이틀 화면에서 확인할 수 있습니다.")

def FavoriteButton(self):
    self.MountainName = self.comboString.get()  # 타이틀에서 산 이름 받아옴
    self.search_word = urllib.parse.quote(self.comboString.get())

    ...
    conn = http.client.HTTPConnection("openapi.forest.go.kr")
    service_key = "cuVGydw6yzwC%2B6YdfYKOPzXxvC45arm%2F1M1dpN31ZrgomglojiWkwCq0jZgneeAvoEZxOqR8WrymypQQvq4hpg%3D%3D
    url = (
        "https://apis.data.go.kr/1400000/service/cultureInfoService2/mntInfoOpenAPI2"
        f"?serviceKey={service_key}&searchWrd={self.search_word}&pageNo=1&numOfRows=10"
    )

    conn.request("GET", url)
    req = conn.getresponse()
    self.tree = ElementTree.fromstring(req.read().decode('utf-8'))

    try:
        if hasattr(self, 'anim_id') and self.anim_id:
            self.Twindow.after_cancel(self.anim_id)
    except Exception:
        pass

    self.Twindow.destroy()  # 기존에 있던 타이틀 윈도우 파괴
    self.InitResult()  # 결과창 생성
```

## “ 즐겨찾기 ”

즐겨찾기 버튼을 누르면 즐겨찾기 리스트에 검색한 산 이름이 추가되고, 재검색 버튼을 눌러 메인 화면으로 돌아가 즐겨 찾는 산을 고르고 즐겨찾기 검색 버튼을 누르면, 해당 산의 정보를 다시 파싱해 산 정보를 확인할 수 있는 윈도우를 생성합니다.

# 04 문제 발견 및 해결

## 지도

- 위도와 경도 정보를 어떤 식으로 받아서 저장할지, 구글 맵 사이트 이미지는 어떻게 그릴지 난감했다.
- 구글 맵을 웹 뷰어로 열고 싶었는데, 웹 뷰어용 윈도우를 여는 것은 어렵지 않았지만 관련 url을 작성하기 힘들었다.

## 이메일

- Tkinter 어플리케이션에서 이메일을 어떻게 전송해야 할 지 감이 잡히지 않았다.
- 전송할 때 어떤 내용을 담으면 좋을 지 아이디어가 필요했다.

↓

- Geocoding API를 사용하여 위도와 경도 정보를 받아 URL 설정 후, pdfcrowd API를 이용해 이미지로 변환했다.
- webview_launcher.py 모듈을 만들어 해당 URL의 웹사이트가 열리도록 했다.

↓

- Gmail SMTP 서버 주소를 이용해 메일을 보낼 수 있었다.
- 산의 상세정보와 저장된 지도 이미지 파일을 보내 시각화 된 정보를 확인할 수 있도록 했다.