**\*과목명: 인프라 활용을 위한 파이썬**

|  |
| --- |
| 기초-1 |
| 모범 답안 |
| import random  def guess\_number\_game():  target\_number = random.randint(1, 100)  attempts = 0  print("1부터 100 사이의 숫자를 맞혀 보세요.")  while True:  attempts += 1  user\_guess = int(input("숫자를 입력하세요: "))  if user\_guess < target\_number:  print("더 높은 숫자입니다.")  elif user\_guess > target\_number:  print("더 낮은 숫자입니다.")  else:  print(f"축하합니다! {attempts}번 만에 숫자를 맞혔습니다.")  break  guess\_number\_game() |

|  |
| --- |
| 기본-1 |
| 모범 답안 |
| import requests  # 실시간 환율 정보를 가져오는 함수 정의  def get\_live\_exchange\_rates():  API\_KEY = "your\_api\_key\_here"  url = f"https://openexchangerates.org/api/latest.json?app\_id={API\_KEY}"  response = requests.get(url)  # JSON 응답으로부터 환율 정보만 추출  rates = response.json()['rates']  return rates  def currency\_converter():  rates = get\_live\_exchange\_rates()  print("사용 가능한 통화:", ', '.join(rates.keys()))    from\_currency = input("변환할 통화(예: USD): ").upper()  to\_currency = input("목표 통화(예: EUR): ").upper()  amount = float(input(f"{from\_currency} 금액을 입력하세요: "))    if from\_currency in rates and to\_currency in rates:  # 입력받은 금액을 from\_currency의 환율로 나누어 기준 통화로 변환한 다음,  # 목표 통화의 환율로 곱하여 최종 환전 금액 계산  converted\_amount = (amount / rates[from\_currency]) \* rates[to\_currency]  print(f"{amount} {from\_currency}는 {converted\_amount:.2f} {to\_currency}입니다.")  else:  print("지원하지 않는 통화입니다.")  # 프로그램을 실행하는 currency\_converter 함수를 호출  currency\_converter() |

|  |
| --- |
| 기본-2 |
| 모범 답안 |
| import os  import time  import re  # 모니터링할 디렉터리 설정  DIR\_WATCH = "static"  # 초기 파일 리스트 저장  previous\_files = set(os.listdir(DIR\_WATCH))  while True:  time.sleep(1) # 시스템 부하 감소를 위한 딜레이  current\_files = set(os.listdir(DIR\_WATCH))  new\_files = current\_files - previous\_files  for filename in new\_files:  print(f"새 파일 생성 감지: {filename}")  file\_path = os.path.join(DIR\_WATCH, filename)  with open(file\_path, 'r', encoding='utf-8') as file:  lines = file.readlines()  for num, line in enumerate(lines):  # 주석 처리된 줄 확인  if line.strip().startswith(("#", "//")):  print(f"{num + 1}번째 줄 주석: {line.strip()}")  # 이메일 주소 검사  if re.search(r'[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{1,}', line):  print(f"{num + 1}번째 줄 이메일 주소: {line.strip()}")  # 파일 리스트 갱신  previous\_files = current\_files |

|  |
| --- |
| 응용-1 |
| 모범 답안 |
| -------------- app.py --------------  from flask import Flask, render\_template, request, flash  from ftplib import FTP, error\_perm  import os  # Flask 애플리케이션 초기화  app = Flask(\_\_name\_\_)  #app.secret\_key = 'your\_secret\_key'  @app.route('/', methods=['GET', 'POST'])  def index():  if request.method == 'POST':  # 폼에서 입력된 데이터 추출  hostname = request.form['hostname']  username = request.form['username']  password = request.form['password']  # FTP 파일 목록 가져오기  file\_list = get\_ftp\_file\_list(hostname, username, password)  # 파일 목록을 index.html 템플릿에 전달하여 렌더링  return render\_template('index.html', file\_list=file\_list)  # GET 요청 시 로그인 폼을 보여주는 login.html 렌더링  return render\_template('login.html')  def get\_ftp\_file\_list(hostname, username, password):  ftp = FTP(hostname) # FTP 서버에 연결  ftp.login(username, password) # 사용자 이름과 비밀번호로 로그인 시도  file\_list = [] # 파일 정보를 저장할 리스트  # FTP 서버의 파일 목록을 가져와 file\_list에 추가  ftp.retrlines('LIST', lambda line: file\_list.append(line.split()))  ftp.quit() # FTP 연결 종료  return file\_list  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True)  --------------------------------------  -------------- index.html --------------  <!DOCTYPE html>  <html>  <head>  <title>FTP 파일 목록</title>  <style>  table {  border-collapse: collapse;  width: 50%;  margin: 20px auto;  }  th, td {  border: 1px solid #dddddd;  text-align: left;  padding: 8px;  }  </style>  </head>  <body>  <h1>FTP 파일 목록</h1>  <table>  <tr>  <th>파일명</th>  <th>크기</th>  <!-- 다른 정보가 필요한 경우 여기에 추가 -->  </tr>  {% for file in file\_list %}  <tr>  <td>{{ file[8] }}</td> <!-- 파일명 -->  <td>{{ file[4] }}</td> <!-- 크기 -->  <!-- 다른 정보가 필요한 경우 여기에 추가 -->  </tr>  {% endfor %}  </table>  </body>  </html>  --------------------------------------  -------------- login.html --------------  <html>  <head>  <title>Login</title>  </head>  <body>  <form method="POST" action="{{ url\_for('index') }}">  <label for="hostname">Hostname:</label>  <input type="text" id="hostname" name="hostname" size="40">  <br>  <label for="username">Username:</label>  <input type="text" id="username" name="username" size="40">  <br>  <label for="password">Password:</label>  <input type="password" id="password" name="password" size="40">  <br>  <button type="submit">Submit</button>  </form>  </body>  </html>  ------------------------------------- |

|  |
| --- |
| 응용-2 |
| 모범 답안 |
| import requests  from bs4 import BeautifulSoup  import smtplib  from email.mime.multipart import MIMEMultipart  from email.mime.text import MIMEText  from email.mime.application import MIMEApplication  import os  import openpyxl  from datetime import datetime  import schedule  import time  SECRET\_ID = "ID" # SMTP 서버에 로그인 할 사용자 ID  SECRET\_PASS = "PASS" # SMTP 서버에 로그인 할 비밀번호  MY\_EMAIL = "MY\_EAIL@naver.com" # 보내는 사람의 이메일  RECIPIENT\_EMAIL = "RECIPIENT\_EMAIL@naver.com" # 받는 사람의 이메일  def mail\_sender(file\_path, now):  try:  # SMTP 서버에 연결 설정  smtp = smtplib.SMTP('smtp.naver.com', 587)  smtp.ehlo()  smtp.starttls()  smtp.login(SECRET\_ID, SECRET\_PASS)  # 이메일 생성  msg = MIMEMultipart()  msg['Subject'] = f"보안 동향 {now} 정보입니다." # 메일 제목  msg['From'] = MY\_EMAIL  msg['To'] = RECIPIENT\_EMAIL  # 이메일 본문 설정  contentPart = MIMEText(f"<html><body><h2>보안 동향 {now} 정보입니다.</h2></body></html>", "html")  msg.attach(contentPart)  # 이메일에 첨부파일 추가  with open(file\_path, 'rb') as f:  etc\_part = MIMEApplication(f.read())  etc\_part.add\_header('Content-Disposition', 'attachment', filename=os.path.basename(file\_path))  msg.attach(etc\_part)  # 메일 전송하고 연결 종료  smtp.sendmail(MY\_EMAIL, RECIPIENT\_EMAIL, msg.as\_string())  smtp.quit()  except Exception as e:  print(f"An error occurred while sending the email: {e}")  def job():  now = datetime.now().strftime("%Y-%m-%d") # 현재 날짜를 문자열로 표현  url = "https://www.malware-traffic-analysis.net/2023/index.html"  headers = {'User-Agent': 'Mozilla/5.0'}  # 엑셀 파일 생성  workbook = openpyxl.Workbook()  worksheet = workbook.active  worksheet['A1'] = "설명"  worksheet['B1'] = "URL링크"  try:  # 웹 페이지에서 데이터 스크래핑  req = requests.get(url, headers=headers)  soup = BeautifulSoup(req.text, "html5lib")  tags = soup.select("#main\_content > div.content > ul > li > a.main\_menu")  row = 2  for tag in tags:  tag\_text = tag.text  tag\_href = f"https://www.malware-traffic-analysis.net/2023/{tag['href']}"  worksheet.cell(row=row, column=1, value=tag\_text)  worksheet.cell(row=row, column=2, value=tag\_href)  row += 1  # 엑셀 파일 저장  workbook.save(f'malware\_{now}.xlsx')  # 이메일 보내기  mail\_sender(f'malware\_{now}.xlsx', now)  except Exception as e:  print(f"An error occurred during job execution: {e}")  # 함수를 스크립트 실행시 즉시 호출  job()  # 스케줄링 설정: 매일 'job' 함수 실행  schedule.every(1).days.do(job)  # 무한 루프를 돌며 스케줄된 작업 실행이 실행될 수 있도록 함  while True:  schedule.run\_pending()  time.sleep(60) # 1분 간격으로 스케줄 확인 |

|  |
| --- |
| 응용-3 |
| 모범 답안 |
| import zipfile  import os  import ftplib  from datetime import datetime  import time  # FTP 서버 접속 정보 직접 설정  ftp\_host = "192.168.100.11"  ftp\_user = "kali"  ftp\_pass = "kali"  def zipdir(path, zip\_file):  for root, dirs, files in os.walk(path):  for file in files:  zip\_file.write(os.path.join(root, file))  if \_\_name\_\_ == '\_\_main\_\_':  directory = 'static'  now = datetime.now().strftime("%Y%m%d")  zip\_file\_name = f'{directory}\_{now}.zip'    # ZIP 파일 생성  zip\_file = zipfile.ZipFile(zip\_file\_name, 'w')  zipdir(directory, zip\_file)  zip\_file.close() # ZIP 파일 명시적으로 닫기    time.sleep(5) # 파일 용량이 크면 딜레이가 발생할 수 있기 때문에 sleep 사용    ftp = ftplib.FTP(ftp\_host)  ftp.login(ftp\_user, ftp\_pass)  f = open(zip\_file\_name, 'rb')  ftp.storbinary("STOR " + zip\_file\_name, f)  f.close() # 파일 명시적으로 닫기  ftp.quit() # FTP 연결 명시적으로 종료 |