1. 인공지능 기초

# 인공지능 개발 환경

본 장에서는 인공지능을 이용하여 실제 문제를 풀기위한 필수적인 학습 사항인 인공지능 개발환경 및 주요 언어에 관하여 학습한다. 특히 실제 현장에서 많이 사용되는 파이썬(Python) 등 주요 인공지능 개발 환경의 설치 및 사용 방법에 관하여 자세히 익힌다. 본 장에서 소개된 인공지능 플랫폼은 후반부의 화공 관련 주요 문제 해결을 위한 언어로서 적절한 수준의 프로그램 및 환경 이해가 반드시 필요하다.

## 개발 언어 소개

### 파이썬

파이썬(Python)은 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)이 개발한 객체지향적 프로그래밍 언어이다. 오픈소스이고 높은 확장성을 가져 신속하고 강력한 개발이 가능하게 한다. 또한, 과학과 공학 및 데이터 분석에서 필요로 하는 많은 기능들을 다양한 라이브러리 형태로 제공하고 머신러닝 및 딥러닝 프레임워크를 지원하기 때문에 인공지능을 위한 표준 언어로 자리잡고 있다.

### 매트랩

매트랩(MATLAB)은 MathWork 사에서 개발한 엔지니가 데이터를 분석하고 알고리즘을 개발할 수 있는 프로그래밍 언어다. 유료 라이센스를 필요로 하지만 활발한 커뮤니티를 토대로 다양한 솔루션을 구현할 수 있다. 행렬을 기반으로 한 계산 기능을 지원하며, 내장된 툴박스와 앱을 통해 여러 알고리즘을 빠르게 적용 및 문서화할 수 있다. 또한 다른 개발 언어와의 연계 및 Simulink를 통한 모델 기반 설계가 가능하여 머신러닝 및 딥러닝에 활발히 응용된다.

### R

R은 오픈소스 프로그래밍 언어로 통계 계산과 그래픽 활용에 특화되어 있다. 벨 연구소의 존 챔버스, 릭 배커, 그리고 앨런 윌크스가 개발한 통계언어인 S에 기반해, 뉴질랜드 오클랜드 대학의 로버트 젠틀맨과 로스 이하카가 지금의 R을 개발했다. 현재는 R 코어 팀이 개발하고 있으며, 다양한 통계 기법과 수치 해석 기법을 지원한다. R 에는 사용자가 제작한 패키지 (추가적인 기능을 하는 함수)를 CRAN (the Comprehensive R Archive Network)으로부터 받을 수 있다. R은 선형 및 비선형 모델링, 통계 분석, 시계열 분석, 분류, 클러스터링 등 다양한 통계 및 그래픽 기술에 강점이 있다.

## 개발 환경 소개

본 장에서는 인공지능 문제를 풀기 위한 개발환경을 소개한다. 풀고자 하는 문제에 따라 필요한 환경만 선택해서 사용하면 된다.

### 파이썬 - Google Colab

Google에서 무료 서비스로 제공하는 Colaboratory(Colab)는 Google 사의 하드웨어에 액세스할 수 있는 주피터 노트북(Jupyter Notebook) 인터페이스를 제공한다. 해당 노트북은 중앙 처리 장치(CPU)로 계산을 수행하거나 특수 그래픽 처리 장치(GPU) 및 텐서 처리 장치(TPU)를 통해 가속화할 수 있는 Google에서 제공 및 유지 관리하는 Linux 기반 가상 머신에서 실행된다.

### Colab 사용하기

먼저, Colab을 사용하기 위해서 구글 계정이 있어야 한다. 이후 Colab 사이트(<http://colab.research.google.com/>)로 접속하여 로그인을 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 1. Colab 시작 화면

로그인하면 지금까지 Colab을 사용해서 작업한 파이썬 파일들을 보여준다. Colab에 처음 접속한 사용자는 아래 그림과 같이 최근 사용 파일이 없을 것이다.

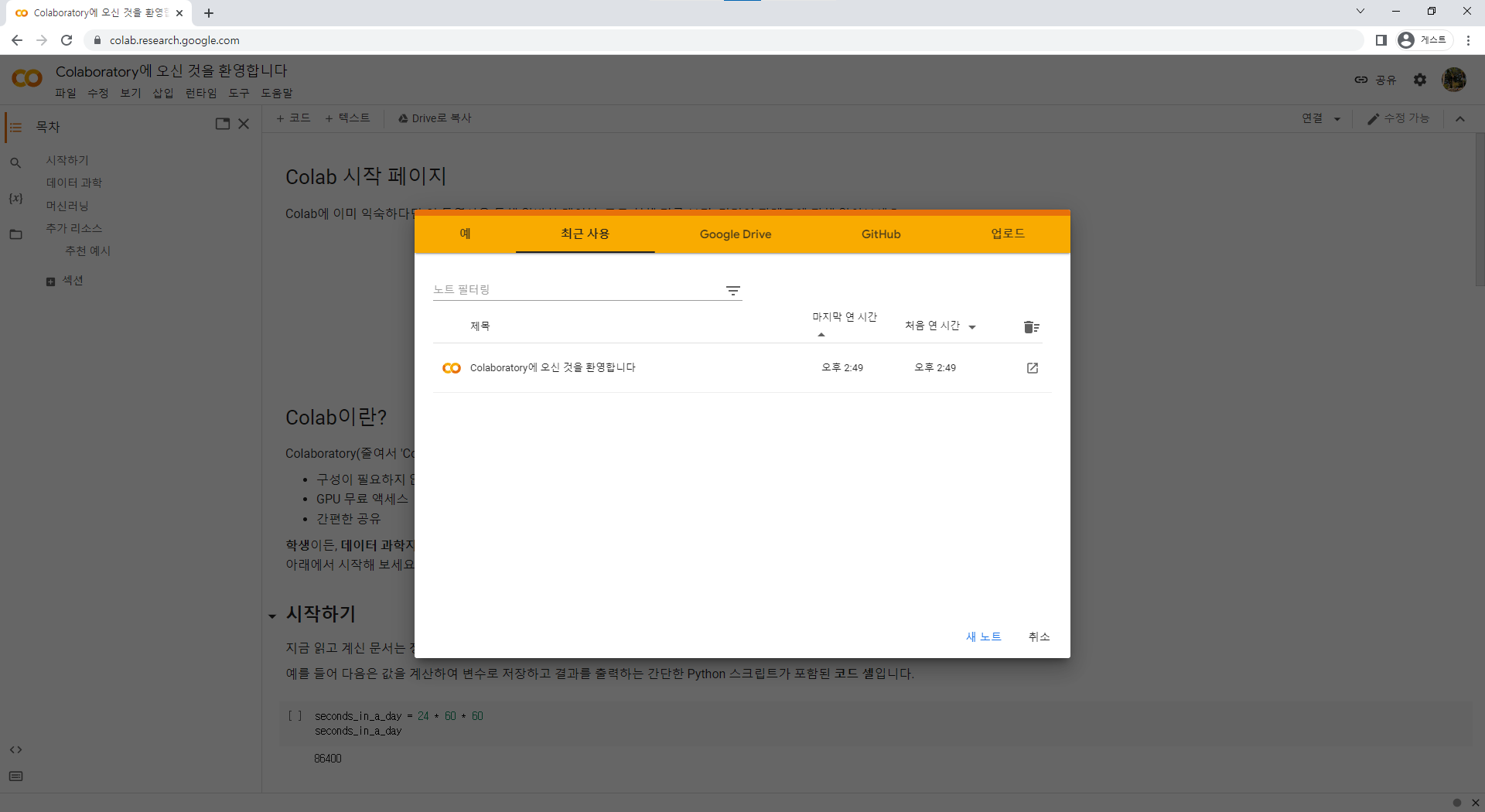


그림 . Colab 최근 사용 파일

파이썬 코딩을 위한 새로운 노트북을 실행하려면 화면 왼쪽 상단에 있는 파일 메뉴를 클릭한다, 이후 하위 메뉴들 중 새 노트 메뉴를 선택하면 된다.

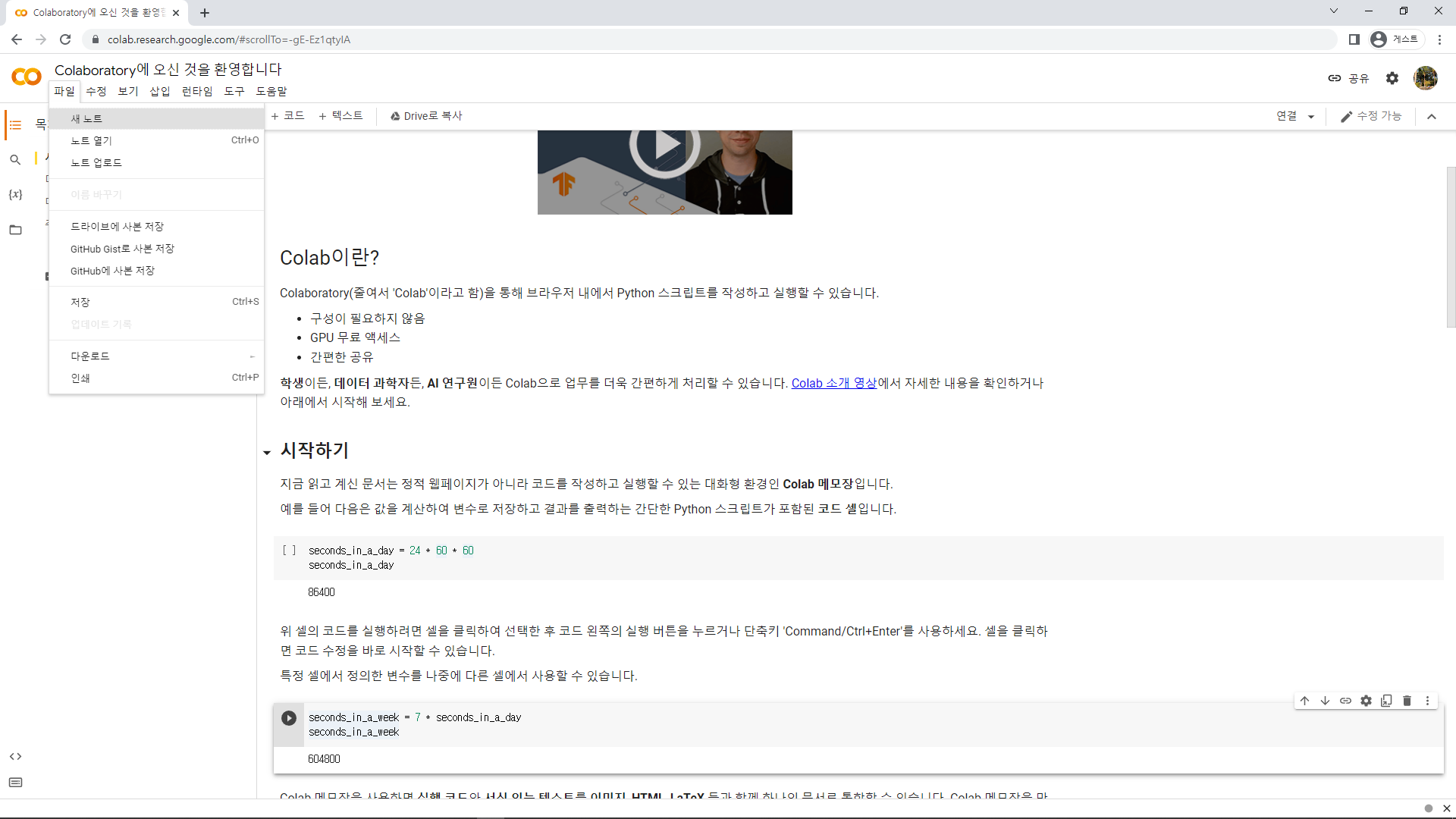


그림 . Colab 새 노트 생성

새 노트를 실행하면 주피터 노트북과 유사하게 파이썬 코드를 입력하고 실행할 수 있는 노트북이 열리고 아래 그림과 같이 코드를 기입하고 실행할 수 있는 셀(cell)이 생성된다.

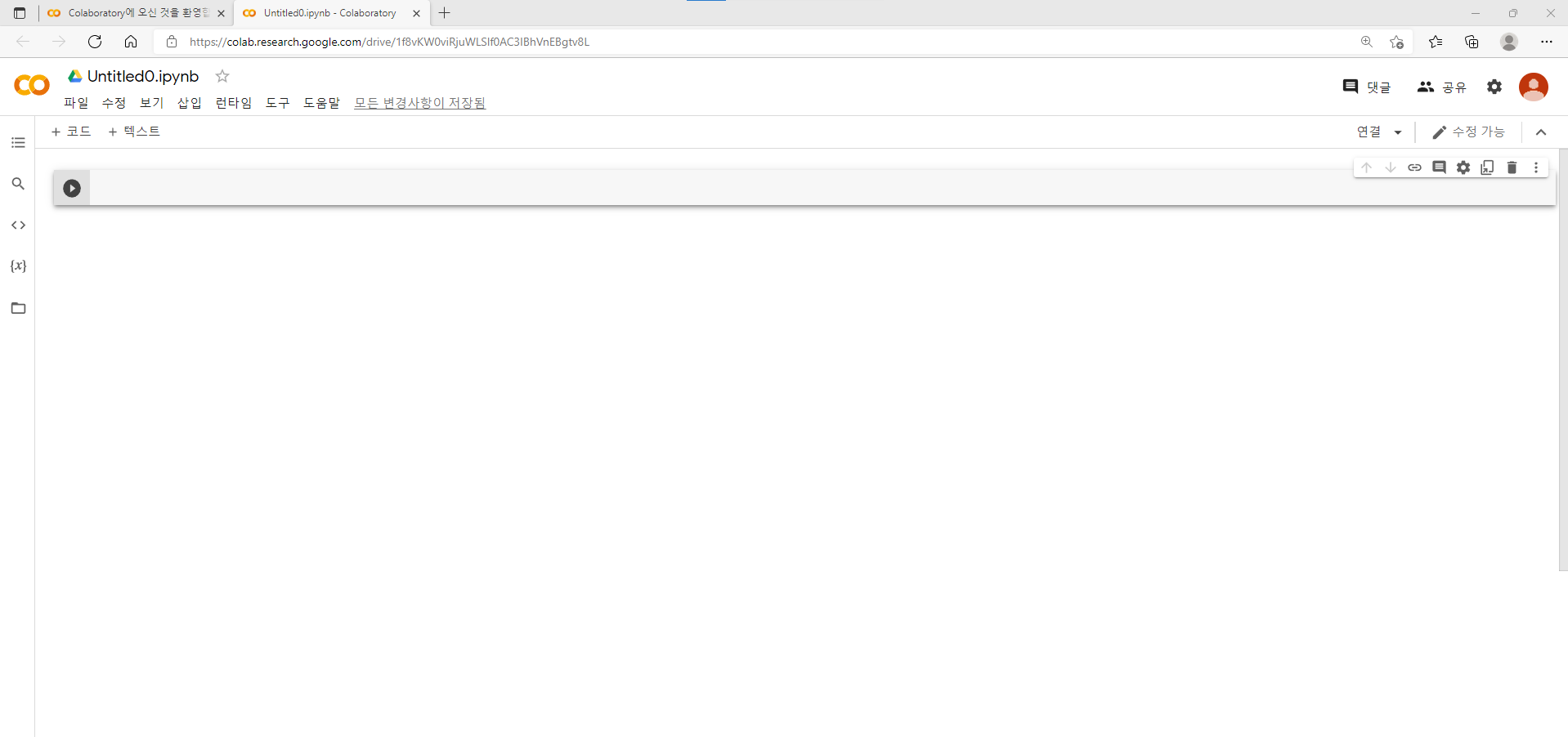


그림 . Colab 새 노트

Colab은 일반적인 개발 환경과 달리 로컬에서 파일을 불러올 수 없고 Google Drive를 이용해야 한다. Google Drive와 연동하기 위해서는 다음과 같은 두가지 중 하나의 방법을 따른다.

방법 1.

1) 왼쪽의 폴더 모양 – 드라이브 마운트 버튼을 눌러준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Google Drive 마운트

2) Google Drive 파일 액세스 허용

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Google drive 연결

방법 2.

1) Google Drive를 마운트 하는 명령어(소스코드)를 입력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Google Drive 마운트 명령어

2) 화면에 표시된 링크로 접속해 연결할 Google Drive 아이디로 로그인하고 허용을 누른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Google Drive 연결 authorization code 입력

3) Google Drive 마운트 허용 코드를 복사하여 Authorization Code 입력 부분에 붙여 넣은 후 연결한다.

### MATLAB

MATLAB은 MathWorks에서 배포하는 수치 계산 및 알고리즘 생성 소프트웨어로 별도의 라이선스가 필요하다. 회사, 대학교 등 각 기관에 라이선스가 없다면 개인을 위한 라이센스를 미리 준비해야 한다.

### MATLAB 설치

MATLAB 설치를 위해 MathWorks 홈페이지로 이동한다.

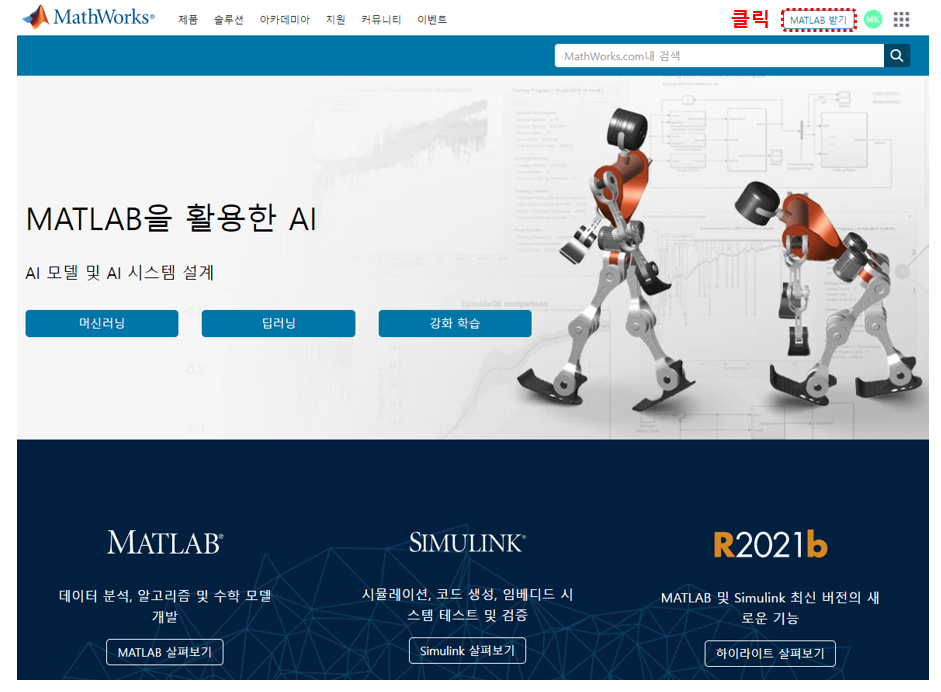


그림 9. Matlab 홈페이지

메인 화면에서 ‘MATLAB 받기’를 누르면 평가판 신청, 캠퍼스용 다운로드, 구매 등이 표시되며 캠퍼스용 소프트웨어를 기준으로 설치를 진행한다.

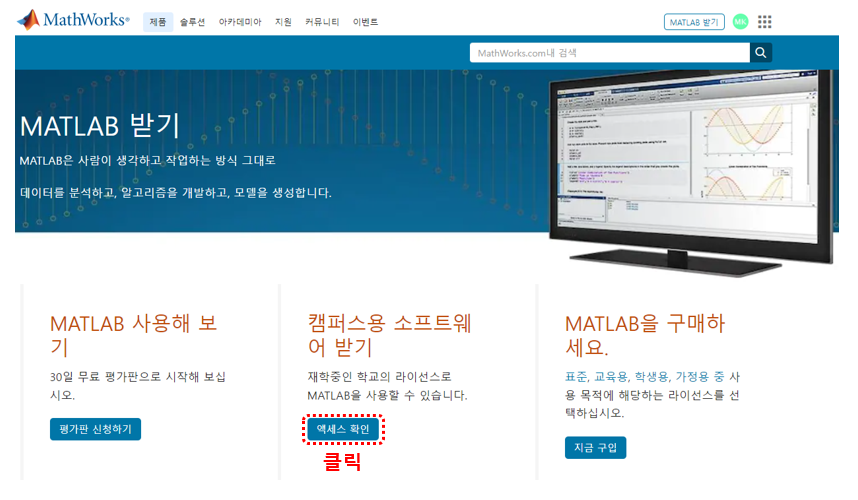


그림 . Matlab 다운로드 페이지

해당 학교 정보와 메일을 입력하면 학교에 등록된 라이선스를 사용할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Matlab 라이선스 입력

해당 기관에 라이선스가 있는 경우 MATLAB 다운로드가 가능한 링크가 보내지며 해당 링크를 통해 설치를 진행한다.

MATLAB에서 지원하는 기능들 중 가장 널리 사용되는 기능들은 기본적으로 다운로드할 수 있도록 체크가 되어 있지만 AI 모델 생성 및 최적화를 위하여 아래 기능들은 반드시 설치해야 한다.

* Deep Learning Toolbox
* Global Optimization Toolbox
* Optimization Toolbox
* Statistics and Machine Learning Toolbox

설치를 마친 후 MATLAB을 실행한다.

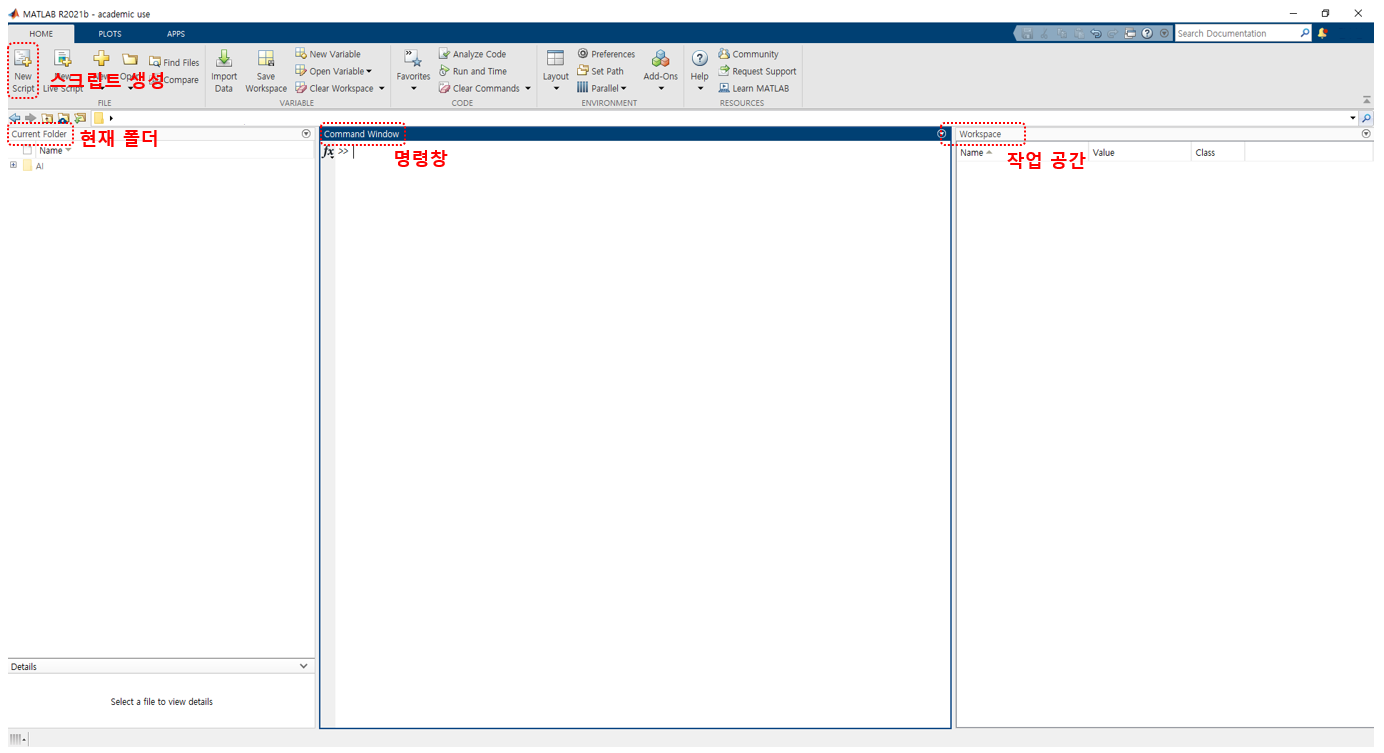


그림 . Matlab 실행 화면

‘현재 폴더’에서 현재 경로에 있는 파일들에 접근할 수 있다.

‘명령창’에서 실시간으로 유저가 작성한 코드를 실행할 수 있다.

‘스크립트 생성’을 통해 어떤 새로운 함수를 구현하거나 코드 실행을 위한 m-file을 생성할 수 있다.

‘작업 공간’은 명령창 또는 스크립트를 통해 실행한 코드에서 사용된 파라미터들에 대한 이름, 값 그리고 형태가 저장된다.

자세한 MATLAB 관련 튜토리얼은 명령창에 help를 입력하거나 ‘Learn MATLAB’을 통해 수행할 수 있다.

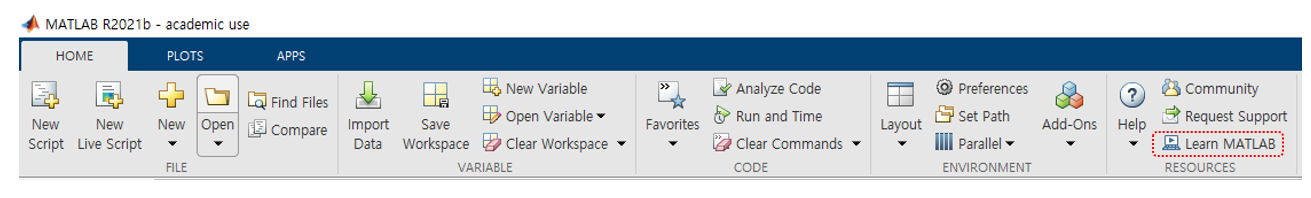


그림 . Matlab 도움말

### R - Rstudio

Rstudio는 R 사용을 위한 통합 개발 환경 (Integrated Development Environment, IDE)로, 어도비 콜드퓨전의 개발자인 JJ Allaire가 개발했다. Rstudio는 일반 데스크톱 어플리케이션과 웹 브라우저 방식으로 사용할 수 있다. 일반적인 IDE와 마찬가지로 디버깅, 컴파일, 배포 등 개발을 위한 기능을 통합적으로 제공한다.

### 2.6. Rstuio 사용하기

Rstudio는 <https://www.rstudio.com/>을 통해 다운받을 수 있다. 일반적인 데스크톱 어플리케이션의 경우는 계정 생성이 필요 없고 무료로 다운받을 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Rstudio 홈페이지

Product tap의 데스크톱 어플리케이션을 선택한다. 일반 Open source edition을 다운로드 받고 설치하면 사용할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Rstudio 다운로드 페이지

Rstudio를 실행하면 아래와 같은 화면을 볼 수 있다. File 탭의 <New File>을 보면 코드 스크립트를 작성할 수 있는 <R script> 버튼이 있다. 우측 상단 Environmental 화면에서는 <Import Datasets> 버튼을 통해 데이터를 환경에 불러올 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 . Rstudio 실행 화면

## 개발 언어 기초 문법

### 파이썬 기초

* 기초자료형(숫자, bool), 변수, 연산자

숫자형은 숫자 형태로 이루어진 자료형을 말한다. 우리가 흔히 사용하는 숫자형으로는 정수, 실수, 복소수가 있고 드물게 사용하는 8진수, 16진수가 있다. 불린 값은 True와 False 중 하나이고, 수학 계산에서는 각각 1과 0으로 평가를 한다.

|  |
| --- |
| >>> a = 267  >>> b = 45.12  >>> c = 0o172  >>> d = 0x8ff  >>> print (3 > 4)  False  >>> print(True or False)  True |

변수 a와 b는 각각 정수형과 실수형을 나타내며, 변수 c와 d는 8진수와 16진수를 나타낸다. 변수를 8진수(Octal)로 선언하기 위해서는 숫자가 0o(숫자 0과 알파벳 소문자 or 대문자 O)로 시작하면 된다. 16진수는 0x로 시작하면 된다.

숫자형은 다양한 연산자를 사용하여 계산을 할 수 있다.

|  |
| --- |
| >>> a = 7  >>> b = 2  >>> a + b  9  >>> a \* b  14  >>> a / b  3.5  >>> a \*\* b  49  >>> b % a  2  >>> a // b  3 |

일반적인 사칙연산(+, -, \*, /)은 한 번쯤은 접했을 연산자일 것이다. 이외에도 제곱을 나타내는 \*\* 연산자, 나눗셈의 나머지 값을 돌려주는 % 연산자, 나눗셈의 몫 값을 돌려주는 // 연산자가 있다.

변수는 값의 이름이다. 알파벳 대소문자 모두 사용 가능하며, 숫자도 변수명의 일부로 사용이 가능하지만 맨 처음에 숫자를 사용할 수는 없다.

|  |
| --- |
| Apple = 451 # 가능  Apple1 = 451 # 가능  2Apple = 451 # 불가 |

* 조건문(if), 반복문(while, for)

프로그램을 작성하면서 제어문을 사용하는 경우가 있다. 제어문은 크게 if, while, for으로 구성된다. 조건을 판단하여 해당 조건에 맞는 상황을 수행하는데 쓰는 것이 if 문이다.

|  |
| --- |
| >>> James = 77  >>> Jessica = 54  >>> if James > Jessica:  … print(‘James is heavier than Jessica’)  … elif James == Jessica:  … print(‘James is equal to Jessica’)  … elif James < Jessica:  … print(‘James is lighter than Jessica’)  …  James is heavier than Jessica |

if 문은 조건식이 True인 경우 명령을 실행하며, False일 경우 명령문은 건너뛰고 다음 코드를 실행한다. if 문은 단일, 2중, 다중 형식으로 구성할 수 있으며, elif 문을 사용하면 이전 조건이 거짓일 때 수행되고 조건을 여러 개 주는 것이 가능하다. 여기서 == 연산자는 일치하는가를 물어보는 연산자이다.

Q1. if문을 사용하여 3개의 숫자 (9, -4, 6) 를 오름차순으로 정렬하여라.

|  |
| --- |
| number1, number2, number3 = 9, -4, 6  if number 1 > number2:  number1, number2 = number2, number1  if number2 > number3:  number2, number3 = number3, number2  if number1 > number2:  number1, number2 = number2, number1  print(“The sorted numbers are”, number1, number2, number3) |

인접한 숫자의 배열이 내림차순인 조건을 if 문으로 선별해 숫자의 배열을 바꾸어 저장해주는 방식으로 숫자의 배열을 오름차순으로 정렬하고 있다.

반복문 while 문은 조건문이 참이 될 때까지 반복해서 실행시킨다. 경험치가 10을 달성하면 레벨이 오르는 게임을 만든다면 다음과 같다.

|  |
| --- |
| >>> EXP = 0  >>> while EXP < 10:  … EXP = EXP +1  … print(“경험치가 %d입니다.” % EXP)  … if EXP == 10:  … print(“레벨업!”)  …  경험치가 1입니다.  경험치가 2입니다.  …  경험치가 8입니다.  경험치가 10입니다.  레벨업! |

여기서 while 문의 조건문은 EXP < 10이다. 10보다 작으면 계속 반복해서 수행하며, EXP = EXP +1은 EXP 값을 계속 1씩 증가시킨다. 그리고 나서 EXP < 10 조건문이 거짓이 되면 while 문을 빠져나가게 된다.

Q2. while 문과 덧셈기호를 이용해 곱셈기호 없이 두 수의 곱셈을 구현해보자.

|  |
| --- |
| m = 3  n = 7  res = 0  while n>0:  res += m  n = n – 1  print(‘m \* n =’, res) |

n 과 m의 곱셈을 구현하기 위해 n이 양수인 조건 하에서 while 문을 반복하며 res 값에 m씩 더하여 저장한다. while 문이 반복될 때마다 n의 값은 1씩 빼서 저장하면, 정확히 n번 반복되는 결과를 가져온다.

다른 반복문으로 for 문이 있다. while 문과 비슷한 반복문인 for 문은 유용하고 문장 구조 이해가 쉽다는 장점이 있다. 다음 for 문의 간단한 예시를 보며 이해를 돕고자 한다. 10명 학생이 운전면허 필기 시험을 보았다. 시험 점수가 70점이 넘으면 합격이고 그렇지 못하면 불합격이다.

|  |
| --- |
| exams = [66, 70, 76, 56, 80, 64, 74, 90, 50, 62]  number = 0  for result in exams:  number = number +1  if result >= 70:  print(“%d번 응시자는 합격입니다.” % number)  else:  print(“%d번 응시자는 불합격입니다.” % number) |

각각의 응시자에게 번호를 부여하기 위해 number 변수를 사용하였다. 리스트 exams에서 차례로 점수를 result 변수로 받아 for 문 안의 문장을 수행한다. 수행이 한 번씩 진행되면서 number는 1씩 증가한다.

Q3. for 문을 이용하여 양의 정수가 소수인지 아닌지 판별해보자.

|  |
| --- |
| n = 9  for a in range (2,n):  if n % a ==0 :  print(‘not prime’)  break;  eles:  print(‘prime’) |

‘for a in range (2,n)’ 문 이하에서 a 자리에 2부터 n-1까지의 수를 넣어가며 한번씩 반복문을 수행한다. N을 2부터 n-1까지의 수로 나눠보았을 때, 하나의 수라도 n을 나누어 떨어지게 하는 수가 있으면 n은 소수가 아니다. 모두 n을 나누어 떨어지게 하지 못한다면 n은 소수이다.

* 함수

프로그래밍을 진행하면 같은 내용을 반복해서 작성하는 경우가 발생한다. 이때 함수를 사용하면 편리하게 작성할 수 있다. 또한, 프로그램 흐름 파악 및 오류가 어디서 발생하는지도 바로 알아차릴 수 있다.

|  |
| --- |
| def multiply(a, b):  result = a \* b  return result |

위는 일반적인 함수의 예이다. multiply는 함수의 이름이며, a와 b는 함수에 사용되는 매개변수이다. 즉, multiply 함수는 2개의 입력 값을 받아 서로 곱한 값을 돌려준다.

다음은 여러 개의 입력 값을 받는 함수에 대해서 알아본다.

|  |
| --- |
| def multiply\_many(\*parameters):  result = 0  for i in parameters:  result = result + i  return result |

위에서의 multiply\_many 함수는 입력 값이 몇 개이든 상관이 없다. \*parameters처럼 매개변수이름 앞에 \*을 붙이면 입력 값을 전부 모아 튜플로 만들어주기 때문이다.

Q4. 두 수의 최대공약수를 구하는 함수를 만들어보자.

|  |
| --- |
| def gcd(a,b) :  while b > 0 :  r = a % b  a = b  b = r  return a |

b가 양수일 조건 하에서, while 문이 반복되고 반복문을 거칠 때마다 a와 b를 b와 r (r은 a를 b로 나눈 나머지)로 업데이트 한다. a를 b 로 나눈 나머지가 0이 될 때, 반복문은 멈추며, 마지막으로 업데이트 된 a 값이 a 와 b의 최대 공약수이다.

* 시퀀스 자료형(문자열, 리스트, 튜플)

시퀀스 자료형에는 문자열, 리스트, 튜플이 있다. 문자열은 문자, 단어 등으로 구성된 문자들의 집합이다.

|  |
| --- |
| >>> a = ‘home’  >>> a[0]  ‘h’  >>> a[1:3]  ‘om’  >>> a[:4]  ‘home’  >>> a[1:]  ‘ome’  >>> a = ‘c’ + a[1:]  >>> a  ‘come’ |

문자열에서는 한 글자마다 번호를 매기게 된다. 맨 앞에서부터 0부터 순차적으로 번호가 매겨진다. a[:4]는 a[4]는 포함하지 않고, a[1:]는 a[1]를 포함하여 출력한다. 시퀀스의 일부를 취하는 것을 슬라이싱이라고 한다.

리스트를 만들 때는 대괄호([])로 감싸주고 각 원소는 쉼표로 구분한다.

|  |
| --- |
| >>> a = []  >>> b = [1, 2, 3]  >>> c = [‘Process’, ‘Engineering’, ‘Systems’]  >>> d = [1, 2, ‘Chemical’, ‘Engineering’] |

리스트는 a처럼 아무것도 포함하지 않을 수 있고, b처럼 숫자를 원소로 가질 수 있다. 또한, c처럼 문자열로 구성될 수도 있으며, d처럼 숫자와 문자열을 함께 가질 수도 있다.

|  |
| --- |
| >>> example = [2, 6, 8]  >>> example.append(4)  >>> example  [2, 6, 8, 4]  >>> example.sort()  >>> example  [2, 4, 6, 8]  >>> example.insert(0, 0)  >>> example  [0, 2, 4, 6, 8]  >>> del example[3]  >>> example  [0, 2, 4, 8] |

위 예를 보면, append는 example에 원소 4를 추가하는 기능하고 sort 함수를 이용하면 example을 원소 크기 순으로 정렬할 수 있다. 여기에 원하는 순서에 원하는 원소를 넣고자 한다면, insert 기능을 사용할 수 있으며, 원소를 삭제하고자 한다면 del 함수를 사용하면 된다.

리스트는 아래처럼 중첩하여 또 다른 리스트를 포함할 수 있다. 이와 같은 형태의 리스트를 중첩 리스트(nested list)라고 부른다.

|  |
| --- |
| >>> my\_list = [“Artificial”, [3, 6, 9], [“Intelligence”, 4, 8]]  >>> print(my\_list[0])  Artificial  >>> print(my\_list[1])  [3, 6, 9]  >>> print(my\_list[1][2])  9  >>> print(my\_list[0][0])  A  >>> print(my\_list[0][2])  t  >>> print(my\_list[2])  [“Intelligence”, 4, 8]  >>> print(my\_list[2][0])  Intelligence |

Q5. 중첩 리스트의 모든 원소들의 합을 구하는 nested\_sum이라는 함수를 만들어보자.

|  |
| --- |
| def nested\_sum(nestedList, newList = [0]):  def flatlist(nestedList):  for I in range(len(nestedList)):  if type(nestedList[i]) == int:  newList.append(nestedList[i])  else:  flatlist(nestedList[i])  return newList  flatlist(nestedList)  print(sum(newList)) |

먼저, 중첩 리스트에 들어있는 리스트 원소들의 포함 원소들을 모두 꺼내 하나의 리스트 안의 원소들로 만들어주는 ‘flatlist’ 함수를 정의하고, ‘flatlist’ 함수를 사용해 중첩 리스트를 하나의 리스트로 만들어 sum 함수를 통해 원소들의 합을 구한다.

마지막으로 튜플은 몇 가지 점을 제외하면 리스트와 비슷하다. 리스트는 []으로 둘러싸이지만 튜플은 ()로 둘러싸인다. 그리고 리스트는 값의 생성, 삭제, 수정이 가능하지만, 튜플은 값을 바꾸거나 삭제할 수 없다.

|  |
| --- |
| >>> s1 = (4, 7, ‘p’, ‘q’)  >>> del s1[0]  텍스트이(가) 표시된 사진  자동 생성된 설명 |

튜플의 원소를 리스트처럼 del을 이용해서 지우려고 하면 위와 같이 오류가 발생함을 알 수 있다. 또한, 튜플의 원소를 변경하려고 한다면 아래와 같은 오류가 발생한다.

|  |
| --- |
| >>> s1 = (4, 7, ‘p’, ‘q’)  >>> s1[0] = ‘a’  텍스트이(가) 표시된 사진  자동 생성된 설명 |

### 매트랩 기초

* 데이터 생성

매트랩에서 사용하는 데이터는 주로 매트랩 외부 데이터를 이용한다. 외부 데이터를 필요한 수식에 적용하기 위해서는 필요한 데이터만을 불러와 원하는 형태로 변수를 지정해 배열하는 것이 중요하다.

Q1. 엑셀 파일(Ex\_data.xlsx)에서 데이터를 불러와 입력 데이터 행렬인 X와 출력(응답) 데이터 벡터인 y를 생성하여라. 이때, X는 엑셀 파일의 무게(Weight), 마력(Horsepower), 실린더(Cylinders), 연식(Model\_Year)으로 구성되어 있으며, y는 마일리지(MPG)로 구성되어 있다.

**텍스트이(가) 표시된 사진

자동 생성된 설명**

데이터는 ‘readtable’ 함수를 통해 불러올 수 있으며, 전체 데이터뿐만 아니라 범위 설정을 통해 필요한 데이터만 불러올 수 있다. 불러온 데이터로 변수를 지정하는 기본적인 방법은 그림 1과 같이 2개의 방법이 있다. 방법 1은 엑셀 데이터의 열 번호를 입력해 변수를 지정하는 방법이고, 방법 2는 열의 이름을 통해 값을 불러오는 방법이다.

* 선형회귀 모델의 생성

회귀 분석은 둘 이상의 변수들 사이의 관계를 나타내는 분석 방법으로, 이를 통해 새로운 1개의 변수를 통해 관련된 다른 변수의 값을 예측할 수 있다. 회귀 분석에는 선형 모델, SVM(support vector machine) 등 다양한 모델이 적용될 수 있다. 아래의 예제를 통해 회귀 분석 모델 중 가장 기초적인 선형회귀 모델을 생성하고자 한다.

Q2. Ex\_data.xlsx를 이용해 자동차의 마일리지를 차의 무게와 마력의 함수로 나타내고자 한다. 마일리지를 y, 차의 무게를 x1, 마력을 x2라고 할 때, 선형모델 객체를 생성하는 내장함수인 ‘fitlm’을 이용하여 선형 회귀 모델을 생성하여라.

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같은 방법으로 성형모델을 생성한다. 이때, X에 입력한 순서대로 x1과 x2가 설정된다. 위 코드를 실행시키면 그림 3과 같은 결과가 생성된다. 생성된 성형 회귀 모델은 ‘y = 47.769-0.0065651\*x1–0.042018\*x2’이다. 이때 각 변수에서 pValue 값이 작은 경우는 그 변수가 출력변수에 큰 영향을 미치지 않는다는 것을 의미한다.

테이블이(가) 표시된 사진

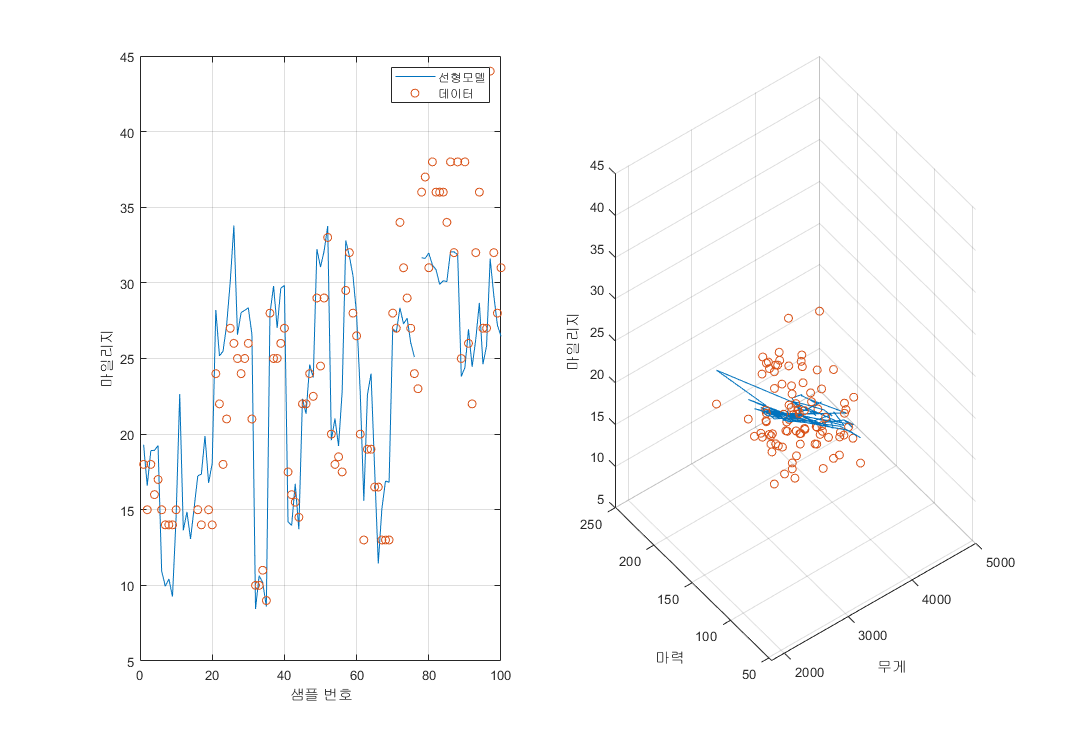
자동 생성된 설명

Q3. 위에서 얻어진 선형모델로부터 얻어지는 마일리지 값과 데이터로 주어진 마일리지 값을 서로 비교하여라.

텍스트이(가) 표시된 사진

자동 생성된 설명

Q2에서 얻은 선형모델과 실제 데이터를 비교하는 그래프를 통해 마일리지 값을 비교할 수 있다. 2차원 비교는 무게와 마력을 하나의 샘플 번호로 합쳐 2차원으로 나타낸 것이고, 3차원 비교는 무게와 마력을 각각 나타내어 3차원으로 나타낸 것이다. 두 그래프를 통시에 나타내기 위해 ‘subplot’ 함수를 사용했다. Subplot 함수는 ‘subplot(m,n,p)’로 나타내며, 이는 현재 그래프를 mＸn 그리드로 나누고, p로 지정된 위치에 좌표축을 만드는 것을 의미한다. 위 코드를 통해 아래 그래프를 얻을 수 있다. 왼쪽은 2차원 비교를 나타내며, 오른쪽은 3차원 비교를 나타낸다.



### R 기초

* 기초 문법

R은 파이썬과 마찬가지로 숫자형, 문자형 등의 자료 형식이 존재한다. “str()” 함수를 통해 다음과같이 데이터 형식을 파악할 수 있다.

|  |
| --- |
| > a = 1  > str (a)  num 2  > b = 2  > str (b)  num 2  > c = ‘k’  > str(c)  chr “k” |

일반적인 사칙연산(+, -, \*, /)은 파이썬과 같은 연산자를 사용한다. 하지만, 나눗셈의 나머지 값은 %% 연산자를 사용한다.

|  |
| --- |
| > a = 7  > b = 2  > a + b  9  > a \* b  14  > a / b  3.5  > a \*\* b  49  > b %% a  2  > a %/% b  3 |

* 조건문(if), 반복문(while, for)

조건문과 반복문은 파이썬과 비슷한 구조를 갖고 있다. 다만, 조건문의 조건 부분과 수행부분이 더 명확히 나눠진다.

|  |
| --- |
| > James = 77  > Jessica = 54  > if (James > Jessica) {  print(‘James is heavier than Jessica’)  } else if (James == Jessica) {  print(‘James is equal to Jessica’)  } else {  print(‘James is lighter than Jessica’) }  [1] “James is heavier than Jessica” |

R에서는 갱신된 정보를 print하기위해 “sprint()” 함수를 사용하며, 해당 정보를 그때마다 저장 (ex. 아래 script의 ‘sys’)하여 “print()” 함수로 불러온다.

|  |
| --- |
| > EXP = 0  > while (EXP < 10){  EXP = EXP +1  sys = sprintf('경험치가 %d입니다.', EXP)  print(sys)  if (EXP == 10){  print('레벨업!')  }}  [1] 경험치가 1입니다.  [1] 경험치가 2입니다.  …  [1] 경험치가 8입니다.  [1] 경험치가 10입니다.  [1] 레벨업! |

for 반복문을 사용하여 파이썬과 마찬가지로 소수를 판별할 수 있지만, 특정 구간의 수를 표현할 때는 “:” 연산자를 사용한다.

|  |
| --- |
| > n = 9  > for (i in 2:8){  if (n %% i == 0){  print('not prime')  } else {  print('prime')  }  }  [1] “prime”  [1] “not prime”  [1] “prime”  [1] “prime”  [1] “prime”  [1] “prime”  [1] “prime” |

## 결론

본 장에서는 인공지능 개발 언어와 개발 환경에 대해 살펴보았다. 풀고자 하는 문제에 따라 언어와 환경을 선택할 수 있다. 그 중에서도 파이썬, 매트랩, R의 기초 문법을 간략히 복습하고 관련 문제를 풀어 봄으로써 추후 실습에서 활용할 수 있다.