DS 부트 캠프 2024 Day1(v1.6)

한상곤(Sangkon Han)
부산대학교 정보컴퓨터공학부
2024/02/05

목차

파이썬 소개 및 개발 환경 설정

파이썬의 특징

- 플랫폼 독립적인 언어
  - Linux, macOS, Windows 등에서 활용
- 인터프리터 언어(interpreter language)
  - 소스코드를 바로 실행, 속도는 느리지만 간편하게 사용 가능 => Jupyter Notebook, VSCode Notebook 등
- 동적 타이핑 언어(dynamic typing language)
  - 실행 시점에서 변수의 타입(type)을 결정, 실행 시점에 메모리 공간을 자유롭게 할당 => 속도는 느리지만 자유롭게 메모리 공간을 할당 받고 사용하는 것이 가능해짐
- 객체 지향 언어 (object oriented language)
  - 명령어 기반이 아닌 정보와 기능을 하나의 독립된 단위로 하여, 각각의 단위가 메시지를 기반으로 문제를 해결하는 방식

데이터 과학/AI에서 파이썬을 사용하는 이유

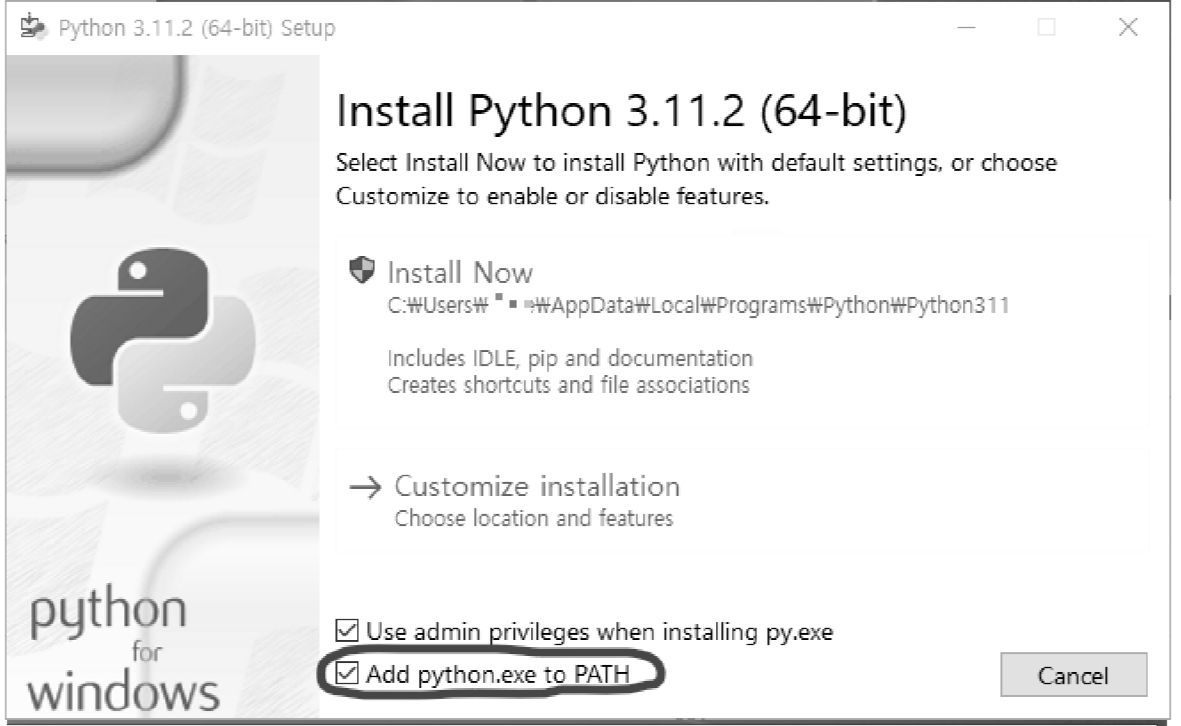
- (상대적으로) 쉽고 간단
  - 다른 언어에 비해서 문법이 단순(no easy, but simple)
- 과학 및 수학 관련 분야에서 두각을 나타냄
  - 아이슈타인의 중력파 증명 실험[1], 태양의 흑점 발견 실험[2] 등에 활용
- 다양한 라이브러리 제공
  - 다양한 라이브러리를 제공, 데이터 과학 및 인공지능 개발을 위한 언어로 인정받고 있음

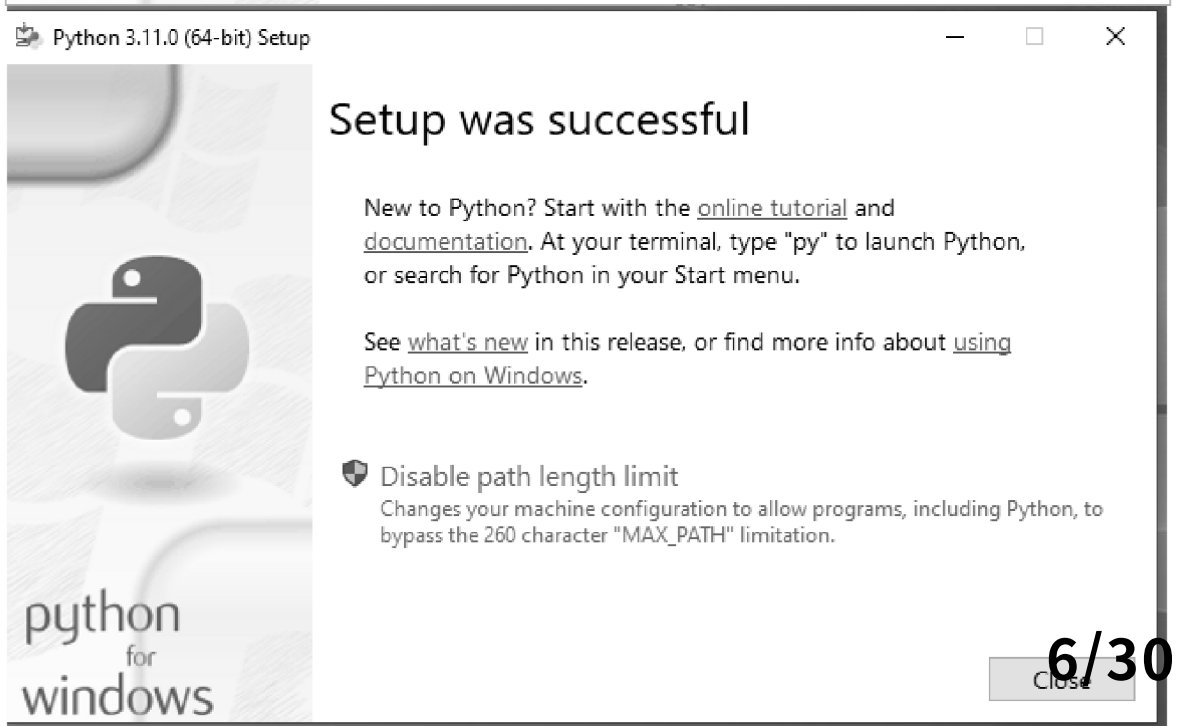
1: 서민호 외, "과학데이터 활용 현황과 수요", KiSTi, 정책보고서, 2016.12.
2: Du Toit, Ruben, et al. "Sunspot identification and tracking with OpenCV." 2020 International SAUPEC/RobMech/PRASA Conference. IEEE, 2020.

개발 환경 설정 - Python

- 어떤 종류의 Python을 설치할 것인가?
  - Python[1] => 일반적인 파이썬
  - {Ana[2], Mini[3]}conda => 과학 및 연구자를 위한 파이썬
- Python 설치시 주의사항
  - PATH(경로) 설정, 경로 길이 최대 설정 해제 등

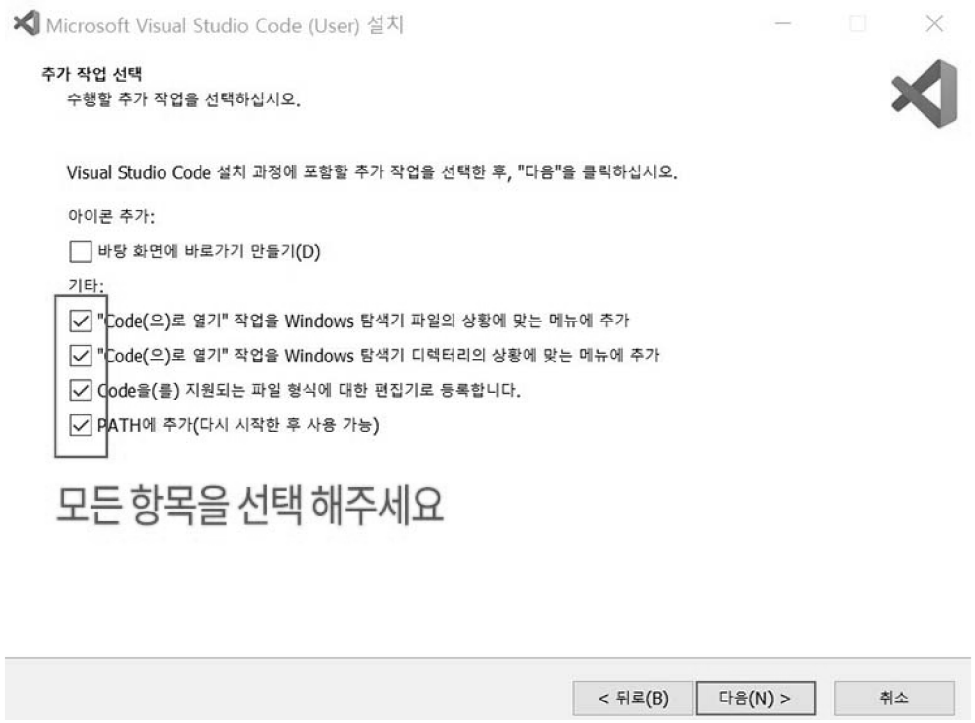
1: https://www.python.org/downloads/release/python-3117/
2: https://www.anaconda.com/
3: https://docs.conda.io/projects/miniconda/en/latest/

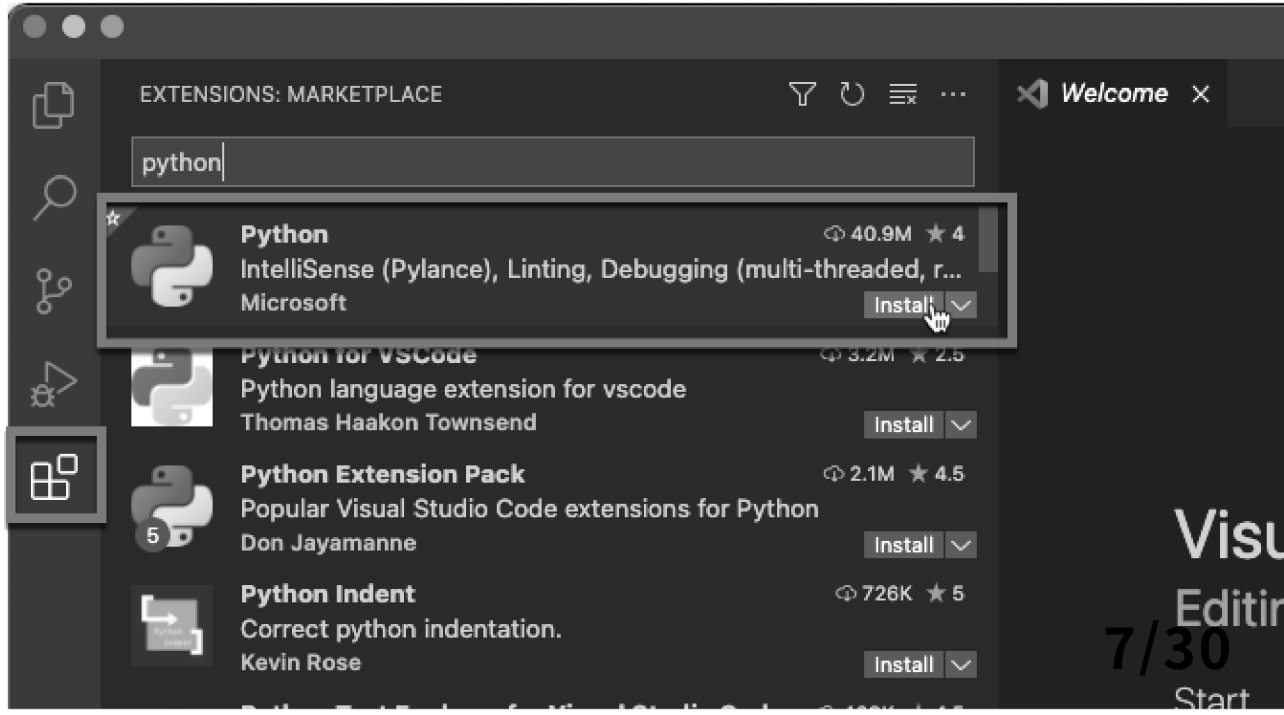




개발 환경 설정 - VSCode

- VSCode 다운로드[1] 후 설치
  - Python 확장 프로그램 설치





1: https://code.visualstudio.com/

변수와 연산자

변수와 메모리

- 변수
  - 수학에서는 변수가 ‘미지의 수’라는 뜻
  - 프로그래밍에서는 ‘어떠한 값을 저장하는 장소’라는 뜻으로 사용됨
- 메모리(memory)
  - 변수를 저장하는 물리적 장소이자 변수에 값이 저장되는 공간
  - 변수는 메모리 주소를 지칭하는 별칭임 => 변수에 들어가는 값은 반드시 어떤 특정한 메모리 주소를 갖게 됨

더 읽어보기 : https://en.wikipedia.org/wiki/Von_Neumann_architecture

변수와 참조

- 변수
  - 메모리의 값을 참조하기 위한 방법
  - 할당 연산자(`=`)를 사용해서 변수에 값을 할당
  - 변수는 값에 대한 참조(`ref`)
  - 변수는 값의 유형(`type`)에 따라 변수의 유형도 정의

```
x = "안녕하세요!" # x라는 변수에 문자열 값을 할당
print(x) # x를 참조하여 값을 출력
print(type(x)) # x의 유형(type)을 출력
```

유형 혹은 자료형

- 데이터 유형
  - 원시 유형
    - `bool`, `int`, `float`, `str`, complex
  - 컨테이너 유형
    - `list`, `tuple`, `dict`, `set`

```
x, y = 5, 2 # x는 5가 할당, y는 2가 할당
print(x + y) # x와 y의 덧셈 결과 출력
s1 = "안녕하세요!" # s1은 문자열 값을 할당
print(s1 + x) # 문자열과 숫자는 덧셈을 할 수 없음
print(x > y) # 비교
```

자료형과 메모리

- 메모리 공간의 기본 개념
  - 변수를 메모리에 저장할 때 그 변수의 크기 만큼 공간을 할당 받음
- 할당 받는 메모리 공간의 크기는 변수의 자료형(data type)에 의해 결정
  - 자연수를 포함해 0, 1, 2, -1, -2와 같이 값의 영역이 정수(`data = 1`)
  - 10.2, 7.2와 같이 소수점이 포함된 값의 영역이 실수(`data = 1.0`)

```
import sys
a, b = 1, 1.0
sys.getsizeof(a)
sys.getsizeof(b)
```

더 읽어보기 : https://en.wikipedia.org/wiki/IEEE_1541-2002

변수 이름

- 가능하면 의미 있는 단어로 표기
- 알파벳, 숫자, 밑줄(_)로 선언
  - `data = 0`, `_a12 = 2`, `_gg = 'afdf'`
  - 변수명은 대소문자가 구분(ABC != abc)
  - 특별한 의미가 있는 예약어는 사용할 수 없음

```
import keyword
print(keyword.kwlist)
['False', ... 'yield']
```

자료형 변환

- 변수의 자료형은 간단히 변환 가능하지만 주의해야 함
  - 자료 손실 => `1.0`에서 `1`로 변환시 발생하는 '정밀도' 손실
  - 문자열 변환시 주의 => 특정 문자 변환시 오류 발생

```
i = 1.0
int(i)
c = "17.i"
float(c)
```

사칙 연산, 정수 그리고 실수 혹은 닫힘성

- 덧셈 연산자: `+`
- 뺄셈 연산자: `-`
- 곱셈 연산자: `*`
- 나눗셈 연산자: `/` or `//`
- 나머지 연산자: `%`

```
25 + 30
30 - 12
50 * 3
30 / 5
30 // 5
30 % 5
```

제어문(flow control)

코드가 실행되는 순서를 제어할 수 있게 해주는 프로그래밍의 기본

조건문(if)

- 조건문(conditional statement)
  - 조건에 따라 특정 동작을 하도록 구성하기 위해서는 기준(상태)과 기준에 따른 방법이 필요
  - 조건문은 반드시 조건의 참(`True`)과 거짓(`False`)으로 구분되어야 함

```
if <조건>:
    T-1
    T-2
else:
    F-1
    F-2
```

조건문(if) 예제

```
print("나이를 입력해 주세요 : ")
my_age = int(input())
if my_age <= 8
    print("OTT KIDz에 오신걸 환영합니다.")
elif (my_age > 8) and (my_age < 19):
    print("OTT Blue에 오신걸 환영합니다.")
else:
    print("OTT에 오신걸 환영합니다.")
```

비교 연산자

- 결과가 `True` / `False`를 반환
  - 파이썬 공식 문서를 꼭 확인
- `is`와 `==`
  - `is`는 `==` 연산자와 다르게 메모리의 주소를 비교

```
a,b = 100,100
a == b # True
a is b # True
a = 300
b = 300
a == b # True
a is b # False
```

연산자 및 각 종 기능에 대한 내용은 가능하면 파이썬 공식 문서를 참고하고, 확인하는 습관을 가집시다. 자신이 사용하는 코드를 작성한 곳에서 발행한 문서를 우선적으로 확인하는 것이 연구를 진행할 때 좋은 습관입니다.

논리 연산자

- `and`, 두 값이 모두 참
- `or`, 두 값 중 하나만 참
- `not`, `True`이면 `False`, `False`이면 `True`

```
a = 8
b = 5
(a == 8) and (b == 4) # False
(a > 7) or (b > 7) # True
not (a > 7)  # False
```

연습1. 학점 계산기

- 90 이상이면 'A'이고, 60 미만이면 'F'이다. 90에서 60사이의 점수는 10점 간격이며 각각 'B', 'C', 'D'이다. `if`, `if-else`, `if-elif-else` 구조를 사용하세요(Hint. if Statements).

```
if score >= 90:
    print("A")
elif (score < 90) and (score >= 80):
    print("B")
elif (score < 80) and (score >= 70):
    print("C")
elif (score < 70) and (score >= 60):
    print("D")
else:
    print("F")
```

반복문(loop)

- 반복문(loop)
  - 문장을 반복하도록 만드는 것으로, 정해진 동작을 반복적으로 수행할 때 사용하는 명령어
- 반복문의 구성
  - 반복 시작 조건, 종료 조건, 수행 명령으로 구성되어 있으며, 들여쓰기와 블록(block)으로 구분
- 조건문에 if라는 키워드가 있듯이, 반복문은 for와 while이라는 명령 키워드를 사용

반복문(while)

- while 문은 조건문이 참인 동안 while 문에 속한 문장들이 반복해서 수행

```
# Fibonacci series:
# the sum of two elements defines the next
a, b = 0, 1
while a < 10:
    print(a)
    a, b = b, a+b
```

```
a, b = 0, 1
while a < 1000:
    print(a, end=',')
    a, b = b, a+b
```

반복문

- for 문은 `in`과 함께 사용되며, `in` 연산자를 활용하여 반복을 진행

```
# Measure some strings:
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

반복문(for)과 range

- 예제1

```
for i in range(5):
    print(i) # 무엇이 출력될까요?
```

- 예제2

```
list(range(5, 10)) # list(range(0, 10, 3)), # list(range(-10, -100, -30))
```

- 예제3

```
a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i]) # 값과 인덱스
```

반복문과 break, else

- 반복문에 break문에 있으면 반복문을 강제로 종료 시킬 수 있음

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
        # loop fell through without finding a factor
        print(n, 'is a prime number')
```

반복문과 continue

- break문과 달리 특정 조건에서 남은 명령을 건너뛰고 다음 반복문을 수행

```
for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found an odd number", num)
```

연습2. 구구단

- 구구단 계산기 프로그램은 사용자가 계산하고 싶은 구구단의 단수를 입력하면 프로그램이 구구단을 출력하는 프로그램을 작성

```
for i in range(2,10):
    print("======", i,"단 ======")
    for j in range(1, 10):
        print(i, "x", j, "=", i*j)
```

과제01 : Anaconda 개발 환경 설정

Anaconda 개발 환경 설정

1. Anaconda를 설치하세요.
2. VSCode를 설치하시고, Python 확장 플러그인을 설치하세요.
3. VSCode에서 Anaconda Python을 설정하세요.
4. `hello.py` 파일 등과 같은 간단한 형태의 파이썬 파일을 생성하세요.
5. 해당 파일에 `print("Hello, World!")`를 입력하시고, 터미널 창에서 `python hello.py`를 실행하세요.
6. 결과를 캡쳐해서 메일로 과제를 보내주세요.