- 3. 파이썬의 이해
  - 1. 파이썬의 개요
    - 파이썬의 탄생 1
      - 히도 판로쉼(Guido van Rossom) 1991년 개발
        - 네덜란드 암스테르담 대학에서 컴퓨터 전공
        - Monty Python's Flying Circus
      - 분산 운영 체제(아메바)의 시스템 관리를 위한 쉘 스크립팅 언어로 개발
        - 쉘 스크립팅 언어 : 절차적 기술
    - 파이썬의 탄생 2
      - 다중 프로그래밍 패러다임 채용
        - 프로그램을 생성하는 접근 방식
        - "명령형 프로그래밍", "절차적 프로그래밍", 객체지향 프로그래밍, 함수형 프로그래밍 지원
      - 다목적 활용
        - 응용프로그램과 웹, 백엔드 개발, 사물 인터넷 분야 뿐만 아니라 교육적인 목적으로 활용
    - 파이썬의 발전 과정
      - 1991년 ABC의 후속 프로그래밍 언어로 시작
      - 1999년 DARPA에 'Computer Programming for Everybody' 제안
      - 2000년 파이썬 2.0 출시
        - 커뮤니티를 통한 개발 체계 시작
      - 2008년 파이썬 3.0 출시
        - 비 하위 호환성을 갖는 메이저 업데이트
    - 파이썬의 인기
    - Pythonic
      - 독립적
        - 특정 회사에 종속X, OS(플랫폼) 독립적
      - 오픈소스
      - 인간적
      - 신속성
        - 라이브러리
      - 직관적
    - 오픈소스
      - 파이썬 관련 개선을 위한 제안(PEP)
        - 많은 개발자의 의견을 수용하고 토론하며 발전한 언어
        - 새로운 파이썬의 기능, 파이썬 프로세스, 환경에 대해 커뮤니티에 설계 문서나 정보를 제공
        - 파이썬 기능의 간결한 기술적 사양과 기능을 위한 근거들을 제공
        - 커뮤니티의 의견을 수집, 합의 도출, 반대의견 청취
      - PEP 8(스타일 가이드) 대표적
    - 인간적 & 직관적
      - 실행할 수 있는 의사 코드(Executable pseudocode) 수준의 문법
        - if 3 in [1,3,5,7] : print("3이 들어있습니다")
          - → 리스트 [1, 3, 5, 7]에 3이 포함되어 있으면 "3이 들어있습니다"를 출력하시오.
    - 생산성 & 신속성
    - 대형 개발자 커뮤니티
    - 라이브러리와 프레임워크
    - 파이썬의 단점
      - C나 자바 등으로 작성된 프로그램보다 느린 속도
      - 완전한 애플리케이션으로 단독 개발이 불가능
        - 쉘 스크립트 언어 용으로 개발
        - 모바일 앱 등 응용 애플리케이션 개발 불가능
          - Rust 또는 Go 고려

- **2.** 파이썬 프로그램의 실행
  - 파이썬 실행 환경
    - **1)** 플랫폼에 독립적이며 **2)** 인터프리터식 **3)** 객체지향적, **4)** 동적 타이핑**(dynamically typed) 5)** 대화형 언어
    - 플랫폼 독립적
      - 윈도우, 리눅스, 유닉스, 맥**OS** 등 다양한 운영체제(플랫폼)에서 별도의 컴파일 없이 실행 가능
    - 인터프리터식
      - **CPython, PyPy, Cython, Jython** 등 다양한 인터프리터 환경 사용 가능
    - 객체지향적
      - 프로그램을 객체로 모델링
    - 동적 타이핑
      - 변수의 자료형을 지정하지 않음
    - 대화형 언어
      - 작성한 코드에 대한 수행 결과를 바로 확인하고 디버깅하면서 코드 작성 가능
  - **CPython**
    - **C** 언어로 개발된 파이썬 인터프리터
      - **C** 구현 라이브러리와의 연동을 통한 확장에 최적
      - 컴파일러의 유형
        - 셀프 호스팅 컴파일러: 부트스트래핑 단계를 통해 자신의 언어로 작성한 컴파일러
        - 소스대 소스 컴파일러 : 타 언어로 작성한 컴파일러
    - 오픈소스로 커뮤니티의 기여로 지속적 발전
  - 파이썬 프로그램 실행과정
    - 파이썬 애플리케이션은 소스 코드 형태로 배포
      - **CPython**이 컴파일 후 바이트코드 **.pyc** 파일 생성
      - 파이썬 가상머신은 바이트코드를 한 라인 씩 실행
      - 변경없이 재실행 시 바이트코드로 빠르게 실행
- **3.** 파이썬 프로그래밍 환경
  - **IDLE**
    - 기본으로 포함된 파이썬의 통합 개발 환경
      - 파이썬과 **Tkinter GUI** 툴킷으로 개발
      - 구문 강조, 자동 완성, 스마트 들여쓰기 등이 포함된 단순한 **IDE** 지향
      - **stepping, breakpoint, call stack**을 확인할 수 있는 통합 디버거 환경 제공
    - 파이썬 공식 홈페이지에서 다운로드 가능
      - **https://www.python.org/**
  - 주피터 노트북
    - 오픈소스 기반의 웹 플랫폼
      - 파이썬을 비롯한 **40**여개의 프로그래밍 언어 지원
      - 전통적인 소스코드-컴파일-실행 방식에서 벗어나 웹 기반 "대화형" 개발 및 실행 환경
      - 문서화하여 다른 사람과 공유하기가 편리
      - 마크다운**(Markdown)**을 이용하여 코드 관련 타이틀, 설명 등 작성 가능
  - 구글 **Colab**
    - **2017**년 과학 연구와 교육을 목적으로 개발
    - 클라우드 기반 주피터 노트북 개발 환경
      - 주피터 노트북 **+** 구글 드라이브를 결합한 서비스
      - 데이터 분석 및 딥러닝 연산 등 고성능 컴퓨팅 리소스 활용 가능
- **4.** 실습 **- Colab**
  - **Google Colab**
    - **https://colab.research.google.com/**
  - **Colab** 이용 방법
    - **1.** 로그인
    - **2. Google Drive**

- + 새노트
- 3. 'chapter3test.ipynb'
  - 3 + 5
    - 실행 버튼을 누르면 GCP로부터 자원 할당
      - 리소스 확인 가능
  - a = 10 print(a)
  - 텍스트 편집 가능
- 4. 설정(톱니바퀴)
  - 행 번호 표시
- 5. 목차
  - 구글 드라이브 마운트