



[모음] 5분 꿀팁 & 단축키

▼ PDF로 다운받기

목차

- 5분 꿀팁
- 프로그래밍은 문제 해결
- 코드 가꾸기
- 용어를 알아보자
- 개발자 마인드 장착하기
- 단축키

5분 꿀팁

프로그래밍은 문제 해결



'프로그래밍은 문제를 해결해나가는 과정'!

모르거나 막히는 부분이 있을 때 여러 가지 방법을 써서 문제를 해결해나가면 됩니다. 개발자들이 문제 해결을 위해 많이 쓰는 방법인 구글링(Googling), 질문 잘하기, 디버깅(에러 해결해나가기) 팁을 드릴게요!

▼ 프로그래밍을 한다는 것은?

- 프로그래밍(programming)은 '문제를 해결하기 위해 컴퓨터에게 명령을 내리는 것'입니다.
- 컴퓨터에게 명령을 내리는 것은 외계인과 대화하는 것과 같죠. 외계인들은 지구인의 문화를 모르기 때문에, 특정한 문법을 따라서, 정확히 코드가 적힌대로만 알아듣습니다.
- 단계별로 정확히 적힌대로만 실행하는게 우리가 평소에 이야기하는 거랑 뭐가 다르냐고요? 😊 영상(5분 28초)으로 확인 하시죠!
 -  THIS "EXACT INSTRUCTIONS CHALLENGE" IS SO HILARIOUS ([링크](#))
 - Youtube 영상 - 한국어 자막으로 보는 방법 : 하단 설정 - 자막 - 자동번역 - 한국어

<https://youtu.be/Ct-IOOUqmyY>

▼ 구글링

- ! 구글링이 무엇인가요?
 - 여러분이 모르는 것, 전 세계 사람들 다 모릅니다. 이럴 땐 필요한 건 구글! 구글에 검색한다고 해서 구글링(Googling)이라고 합니다.
- ! 구글(Google)에만 검색해야 하나요? 네이버, 다음은 안되나요?
 - 구글로 검색하면 국내 블로그, 사이트 중심이 아니라 전 세계 풍부하고 방대한 자료가 나옵니다. 개발은 전 세계 사람이 하니까 보다 많은 자료를 접할 수 있겠죠?
- 아래 내용을 나중에 스스로 읽어보면서, 구글 검색 잘하는 방법을 구글링해보세요! 😊
 - 추천 키워드 : 구글링하는 법!, how to googling for programmer, 구글 검색 팁
 - 검색 결과 예: 개발자를 위한 Google 검색 노하우 - 김환규 hosted by OKKY ([1편 링크](#) / [2편 링크](#))

▼ 검색어 팁

- 아래처럼 검색어를 조합해 입력해보세요!

- 기술을 처음 배우고 싶을 때 : '기술이름' + 'tutorial' (예. `Javascript tutorial`)
- 기능을 찾을 때 : '기술이름' + 'how to' + '찾을 내용' (예. `Javascript how to hide div`)
- 어떻게 사용하는지 예제를 보고 싶을 때 : '기술이름' + '내용' + 'example' (예. `Javascript onclick tutorial`)
- 원하는 사이트명 포함해 검색할 수도 있습니다. (예: `stackoverflow Javascript how to hide div` - stackoverflow 라는 사이트에서 검색)

▼ 검색 결과 중에, 좋은 자료 고르기

- 좋은 자료를 찾으려면 경험치가 필요해요. 많이 검색해보면 자연스레 나만의 검색 노하우와 자료 판단하는 눈이 길러질 거에요.
- 검증과정을 거쳐 잘못된 내용이 금방 수정되거나 오류 자체가 적은 사이트
 - MDN([링크](#)) - 참고. MDN을 신뢰할 수 있는 이유([링크](#))
 - 기술 공식 문서(예를 들면, [부트스트랩 컴퍼넌트 페이지](#))
 - 신뢰할 수 있는 블로그(tech 회사의 기술 블로그, IT 전문 매거진)
 - stackoverflow([링크](#)) - 개발 QnA 사이트입니다. 전 세계적으로 많이 쓰입니다. 질문과 답변, 댓글에 사용자들이 vote 할 수 있어요. 좋은 질문과 답변에는 vote 수가 높습니다. ([링크](#))
- 그 외에
 - 해결책뿐만 아니라 문제(에러)의 이유까지 적어두어서 내 문제와 같은지 판단할 수 있는 정보를 제공하는 글
 - 오래되지 않은 자료 - 웹의 경우, 기술이 빠르게 발전하기 때문에, 몇 년이 지난 오래된 자료는 버전 등의 문제로 내용이 달라질 수 있어요.

▼ 검색 결과글에 모르는 용어가 너무 많다면?

1. 자, 일단 심호흡합시다. 침착하세요! 내가 이해할 수 있는 만큼만 부분부분 찬찬히 읽어봅시다.
2. 읽어보고 내가 따라서 하는게 가능한가요? 그럼 한 번 따라서 해보죠! 부분적으로 안되거나, 미심쩍은 부분, 더 알고 싶은 내용은 슬랙 채널에 질문해보세요. (단, 참고한 자료 링크도 포함해서 질문해야겠죠?)
3. 따라하기 어렵나요? 내용을 전혀 알 수가 없나요? 좋아요. 그럼 다른 검색결과로 넘어가죠! 내 수준에 맞는 자료를 찾으면 됩니다.
- 이렇게 해도 되냐고요? 그럼요. 우리는 프로그래밍을 배우고 있는 과정이니까 모르는 내용이 많아도 괜찮아요. 처음부터 끝까지 하나도 빠짐없이 꼭 알아야만 하는 건 아니랍니다.

▼ 만약 검색 결과가 잘 나오지 않는다면?

- 여러 번 검색하기 : 처음에는 문제를 잘 모르기 때문에, 검색하면서 정보를 얻게 됩니다. 얻은 정보를 가지고 검색어를 수정해보세요.
- 검색어 자동완성 & 연관검색어 사용 : '자동완성 되는 검색어' 와 검색 결과 하단에 보이는 '연관 검색어'로 검색해보세요. 구글에서는 현재 검색 결과에서 사람들이 많이 검색한 검색어와 결과가 많은 검색어를 추천해줍니다.
- 검색어 검토하기 : 한 발 물러서서, 내가 입력한 검색어를 살펴보세요. 너무 검색어 범위가 넓지 않은지 / 여러가지 해결책 중에 내가 생각한 특정 해결책으로만 검색하고 있지 않은지!

▼ 질문 잘하기

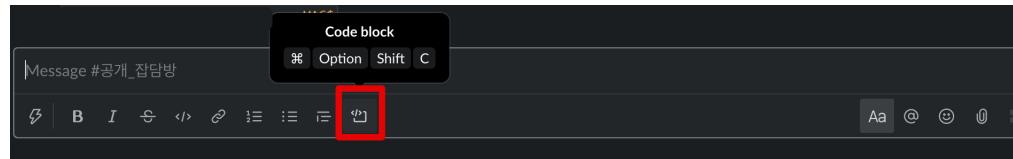
- 세상에 바보같은 질문은 없습니다! 하지만 좀 더 잘 질문할 수는 있죠!
- ! 질문 그냥 하면 되는 거 아닌가요? 제가 아무렇게나 이야기해도 잘 알아들어야죠!
 - 질문을 기술적인 개념을 설명하고 커뮤니케이션하는 연습 기회로 삼아보세요.
 - 개발은 협력해야하는 것들이 많습니다. 혼자 처음부터 끝까지 만드는 경우는 매우 드물어요. (우리도 부트스트랩 같은 외부 자원을 가져다 사용하죠)
- 어떻게 질문하느냐에 따라 심지어 답변을 못 받기도 해요. 사람들이 이해할 수 없는 질문에 어떻게 답변하긴 어렵겠죠.
- 질문 잘하기! 네 하지만 기억하세요!

▼ 1. 내가 해결하고 싶은 문제를 정확히 알려주기

- 여러분들이 고민하고 있는 코드를 올려주세요. 그리고 에러가 나는 라인수(예. 180번째줄)을 정확히 알려주세요.
- 정확히 이 부분이라고 아는게 아니면, 되도록 설정과 관련된 부분과 앞 뒤 관련된 부분을 모두 올려주셔야합니다! 문제의 종류는 다양합니다. 앞 부분 때문에 발생하는 에러도 있고 import 에러도 있고요. 부분만 봐서는 파악하기 어려운 경우가 많아요.
- 수업 관련 질문은 파일 통째로 올려주기 😊
- 에러 메시지 원본 그대로
- 에러 메시지는 스크린캡쳐보다 텍스트 메시지로 올리는게 좋아요. (화면 모양과 같이 이미지로 봐야하는 건 제외)
- **5분 꿀팁 - 구글링** 편에서 에러메시지로 구글링 하는 거 배웠었죠? 이미지 파일이면 복사-붙여넣기로 검색하기가 어렵겠죠?

▼ 참고. 슬랙 메시지에서 코드 붙여넣는 법

1. 짧은 코드일 경우
 - `(백틱)으로 감싸기 예. '짧은 코드'
2. 긴 코드일 경우
 - `'''(백틱 세 개)로 감싸거나
 - 메시지창 버튼 사용하기



▼ 2. 지금 내가 무엇을 알고 있는지 알려주기

- 현재 수준 알려주기
 - 개발 커뮤니티처럼 사람들이 여러분의 정보를 모를 때에는 '제가 python 함수까지 배웠습니다.' 라고 정보를 알려주면 여러분의 현재 수준을 고려해서 답변을 하기 쉬워요.
- 구글링해서 찾고 이해한 링크
 - 앗, 설마 구글링으로 찾아보지도 않고 그냥 바로 질문하는 건 아니겠죠? **개발할 때는 스스로 문제를 해결하기 위해 고민하면서 경험치가 많이 쌓입니다. (a.k.a 삽질)**

▼ 3. 어디에 어떤 시간에 질문할지

- 질문은 되도록 공개적으로 하는 걸 추천드립니다. 1:1 질문에 대한 지식은 두 명에게만 남지만, 공개적으로 하는 질문에 대한 지식은 다수에게 남겠죠.
- 사람들이 많이 질문하는 곳을 몇 군데 알려드릴게요.
 - 스파르타코딩클럽 슬랙채널 🔥
 - stackoverflow([링크](#)) : 개발 QnA 사이트
 - 페이스북 /슬랙 개발자 커뮤니티
 - Javascript 커뮤니티에서 Python 질문하면 사람들이 어랏? 하겠죠?
 - 기술별(Javascript, Python,...)로, 직군별(프론트엔드, 백엔드, 인프라,...)로 커뮤니티가 세분화되어 있기도 합니다.
 - 기술과 관련된 Github repository 의 issue ([링크](#))
 - 나만 겪는 게 아닌 거 같은 버그일 때 여기에 버그알림(bug report)를 해주시면 좋겠죠?
 - repository 마다 분위기가 조금씩 다르니 기존에 있었던 issue를 보면서 파악해보세요

▼ 4. 질문 해결방법에 대해 피드백하기

- 시간을 들여 답변을 해준 사람에게 **감사합니다!** 인사 한마디 남기는 거 너무 당연하겠죠?

- 답변받은 방법대로 고쳐졌는지, 그렇지 않은지도 정보가 될 수 있어요. 여러분과 같은 고민을 하고 있는 사람이 이 방법대로 하면 해결되는지, 안되는지 알 수 있겠죠?

한 걸음 더!

- 스파르타코딩클럽 슬랙에 여러분들이 알고 있는 내용 질문이 올라온다면, 댓글로 답변해보세요. Learning By Teaching 가르쳐주면서 배우기!
- `질문하는 법`, `how to ask` 키워드로 구글링해보기

▼ 디버깅(Debugging)

프로그래밍을 시작하자마자, 우리는 생각했던대로 프로그램 하는 것이 쉽지 않다는 것을 알게 되었죠. 디버깅은 발견해내야만 하는 것입니다.

저는 정확한 순간을 기억합니다. 내 인생의 많은 부분을 내가 만든 그 프로그램에서 실수를 찾는데 쓸 것이라는 것을 깨달았을 때를.

— 초기 컴퓨터 EDSAC 개발자 Maurice Wilkes 교수,
1979년 9월 23일 "The Design and Use of the EDSAC," 강의 중에서

**이 버그를 고치는 데 긴 시간이 걸린다면 왜 그런지 자문하라.
다음 번에는 이 버그를 좀 더 쉽게 고칠 수 있도록 할 수 있는 뭔가가 있을까?**

— 앤드류 헌트, 데이비드 토머스, 『실용주의 프로그래머』, 김창준, 정지호 옮김, 인사이트(2005), p167

- **?** 디버깅(Debugging)이 무엇인가요?
! 디버깅은 버그를 찾아내어 고치는 과정입니다.
버그(bug)는 소프트웨어가 오작동하거나 예상치 못한 잘못된 결과를 내고, 오류가 발생하는 등의 문제를 말하죠. 설계(Design)가 잘못되었거나, 프로그램 소스 코드에서 오류가 났기 때문에 생기죠.
! 프로그램 만들기는 설계(Design), 구현(Coding), 테스팅(Testing), 디버깅(버그 찾기, Debugging) 과정을 거치게 됩니다. 그만큼 디버깅은 프로그래밍할 때 빼먹을 수 없는 중요한 요소죠.
- 이제 여러분들이 디버깅의 세계  입문용 간단한 가이드를 드리겠습니다.
인생은 실전! 앞으로 실습하고 내 프로젝트를 진행하면서 디버깅을 마음껏 해보세요!

▼ 1. 당황하지 말고 잠시 시간을 두기

- 버그를 발견하면, 잠시 멈춰서 버그를 어떻게 하면 빠르게 찾을 수 있을까? 내가 맞닥뜨렸던 버그인가? 하고 생각해봅시다. 이 경험이 쌓이면, 디버깅을 조금 더 효율적이고 빠르게 할 수 있을 거예요.
- 먼저 심호흡 합니다. 버그는 어디서나 발생할 수 있고, 디버깅은 프로그래밍 하는 과정이니 일단 편하게 마음을 가집니다.

▼ 2-1. 에러 메시지 읽어보기

- 버그가 발생한 부분에 에러 메시지가 있다면 찬찬히 읽어봅니다.
- 이미 에러 메시지가 많은 정보를 알려주고 있는 경우가 많습니다. 왜 에러가 발생했는지 이유와 함께 에러가 발생한 코드의 라인 수를 같이 보여주는 경우가 많습니다. 해당 라인으로 가서 앞 뒤 코드를 읽어봅니다.
- Javascript에서 에러 메시지는 웹 브라우저의 개발자도구 console창에서 확인할 수 있습니다. 많이 발생하는 에러 메시지들은 아래와 같습니다.

Top 10 Most Common JavaScript Error Messages - dummies

To help you decipher the error messages that JavaScript throws your way, here's a list of the ten most common errors and what they mean: Syntax error: This load-time error means that JavaScript has detected improper syntax in a statement. The error message almost always tells you the exact line and character where the error ...

[d https://www.dummies.com/web-design-development/top-10-common-javascript-error-messages/](https://www.dummies.com/web-design-development/top-10-common-javascript-error-messages/)

- Python에서 에러 메시지는 PyCharm 하단 Run 창에서 확인할 수 있습니다. (Traceback으로 시작하는 메세지)

Understanding the Python Traceback - Real Python

Python prints a traceback when an exception is raised in your code. The traceback output can be a bit overwhelming if you're seeing it for the first time or you don't know what it's telling you. But the Python traceback has a wealth of information

<https://realpython.com/python-traceback/>



9.2 오류를 고치는 방법 | 파이썬 프로그래밍 입문서 (가제)

오류를 찾아 고치려면 추리소설 탐정처럼 하면 된다. 오류가 발생한 현장(코드)를 꼼꼼히 살펴보고, 프로그램이 죽기 전에 남긴 오류 메시지를 해석하고, 로그 기록에 따라 프로그램의 실행 흐름을 추적하고, 테스트 환경을 꾸며 오류를 재현해보면 대부분의 오류를 해결할 수 있다. 버그와 디
<https://python.bakyeono.net/chapter-9-2.html#922-%EC%98%A4%EB%A5%98-%EB%A9%94%EC%8B%9C%EC%A7%80>



▼ 2-2. 에러 메시지가 없는데 내가 원하는 결과와 다르다면

- 버그가 발생한 부분에 에러 메시지가 없고, 내가 원하는 결과와 다르게 작업이 되었다면 직접 단계별로 데이터가 어떻게 변하는지 확인해야합니다.
- 또 버그가 발생한 상황을 그대로 만들어보는게 중요하죠. 이런 것을 '버그 재현'한다고 표현합니다.
- 여러가지 방법이 있지만 우리가 배우는 단계에서는 단계별로 데이터가 어떻게 변하는지 값을 '출력해서 디버깅하기(Debugging by Printing)'가 유용합니다. 아래 단계에서 확인해볼게요.

▼ 3. '출력해서 디버깅하기(Debugging by Printing)' 와 디버거(Debugger)

- 버그가 발생한 부분이 명확하지 않다면 소스 코드에서 어떤 단계까지 제대로 동작하고 어떤 단계부터는 제대로 동작하지 않는지 찾아내야합니다.
- 이때 사용하는 것이 '출력해서 디버깅하기(Debugging by Printing)' 와 디버거(Debugger)입니다.
- 출력해서 디버깅하기(Debugging by Printing): 내가 컴퓨터에게 명령어를 내려서, 내가 가진 데이터가 바뀔 때 마다 그 데이터를 출력해보는 것입니다. 어떤 부분까지 내 의도대로 동작했고, 어떤 부분부터 원하는 대로 동작하지 않았는지 확인할 수 있죠.

디버거(Debugger)를 사용하면 이런 동작을 조금 더 편리하게 할 수 있습니다.

- 디버거(Debugger): 어느 부분에서 버그가 발생했는지 쉽게 파악하기 위한 도구입니다. 우리가 사용하고 있는 크롬 개발자 도구와 PyCharm에도 디버거가 있습니다.

▼ Chrome Debugger 튜토리얼

Get Started with Debugging JavaScript in Chrome DevTools

This tutorial teaches you the basic workflow for debugging any JavaScript issue in DevTools. Read on, or watch the video version of this tutorial, below. Finding a series of actions that consistently reproduces a bug is always the first

<https://developers.google.com/web/tools/chrome-devtools/javascript>



▼ PyCharm Debugger 튜토리얼

디버거 - 기능 | PyCharm

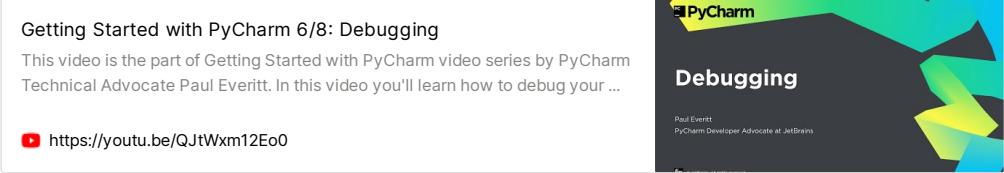
PyCharm은 Python, virtualenv, Anaconda 또는 Conda env와는 관계없이 로컬 컴퓨터에서 실행되는 코드를 디버그할 수 있습니다. PyCharm Professional Edition의 경우 Docker 컨테이너, VM 내에서 또는 SSH를 통해 원격 호스트에서 실행되는 코드도 디버그

[JB https://www.jetbrains.com/ko-kr/pycharm/features/debugger.html](https://www.jetbrains.com/ko-kr/pycharm/features/debugger.html)

PyCharm

The Python IDE
for Professional Developers





▼ 4. 질문하기 - 새로운 관점으로 보기

- 오랫동안 고민해도 버그가 보이지 않는다면, 터널비전(터널 속으로 들어간 것처럼 시야가 극도로 좁아지는 것)에 빠진 것일 수도 있어요. 내 눈에는 안 보였던 문제가 다른 사람에겐 금방 보일 수도 있지요.
- 이럴 때 새로운 관점이 필요합니다. 3주차 5분 꿀팁에서 배운 **질문 잘하기**를 참고해 스스로에게, 다른 사람에게 질문해보세요!

☞ [모음] 5분 꿀팁 & 단축키

▼ 참고. 비한국어 자료를 한국어로 보고 싶을 때 - 한글 자동번역

- 유튜브 영상 자막 한글 자동번역 <https://support.google.com/youtube/answer/6373554?hl=ko>
- 크롬 자막 번역 확장프로그램 <https://chrome.google.com/webstore/detail/google-translate/aapbdbdomjkkjkaonfhkkikfgjllcleb?hl=ko>
- 앞으로 더 프로그밍을 하게 되면, 로깅(logging) - 프로그램 실행되는 과정을 나중에도 볼 수 있도록 로그로 기록 합니다.

코드 가꾸기

▼ 코드 가꾸기 - 코드에 주석 달기

- ! 주석은 언제 사용하나요?
 - 1) 필요없어진 코드를 삭제하는 대신 임시로 작동하지 못하게 하고 싶을 때
 - 2) 코드에 대한 간단한 설명을 붙여두고 싶을 때 사용합니다.
- 주석을 붙여놓으면, 브라우저/컴퓨터가 읽지 않아요. 즉, 개발자 본인 또는 동료를 위해 붙여두는 것!
- Pycharm에서 적용하기 : 주석처리하고 싶은 라인들을 선택(블록처리) → **ctrl(또는 command) + / (슬래시)**

▼ 코드 가꾸기 - 코드 정렬하기

- ! 정렬이 왜 중요한가요?
 - 코드 정렬이 제대로 되어있지 않으면, 코드의 생김새가 한 눈에 들어오지 않아요. 오류(버그)를 해결하기가 무척 어려워집니다.
 - Pycharm에서 적용하기: 정렬하고 싶은 파일에 커서를 두고 → **control(또는 command) + alt(또는 option) + L**

▼ 코딩 컨벤션 - 사람이 이해할 수 있는 코드를 위한

컴퓨터가 이해할 수 있는 코드는 어느 바보나 다 짤 수 있다.

좋은 프로그래머는 사람이 이해할 수 있는 코드를 짠다.

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

— Martin Fowler, 『Refactoring』, Addison-Wesley Professional June(1999), p35

? 코딩 컨벤션(Coding Convention)이 무엇인가요?

! 코딩 스타일을 포함한 규약(조직체 안에서, 서로 지키도록 협의하여 정하여 놓은 규칙)입니다. 코딩 컨벤션은 사람을 위한 약속이라고 할 수 있어요. 사람의 이해를 돋고 읽고 관리하기 쉬운 코드를 만들기 위해서 존재합니다.

맞춤법에 맞는 글은 훨씬 보기 좋죠? 코드도 마찬가지입니다. 코드가 일관성 있는 스타일을 가지고 있다면, 읽기도 좋고 코드의 의도를 파악하기도 편합니다. 가독성이 좋아져 버그를 발생시킬 수 있는 코드를 발견할 확률도 높아지죠. 더불어 다

른 사람들과 협업하기에도 좋습니다.

프로그래밍 언어마다, 회사마다 코딩 컨벤션을 가지고 있어요. 우리가 JS를 배울 때에 사용한 camelCase(단어가 연결되는 첫 글자를 대문자로 함)하고, Python에서는 snake_case(_ 로 단어 연결)를 사용한 것도 코딩 컨벤션이죠.

▼ Javascript 코딩 컨벤션들

- 코딩 스타일 소개

코딩 스타일

개발자는 가능한 한 간결하고 읽기 쉽게 코드를 작성해야 합니다. 복잡한 문제를 간결하고 사람이 읽기 쉬운 코드로 작성해 해결하는 것이야말로 진정한 프로그래밍 기술입니다. 좋은 코드 스타일은 이런 기술을 연마하는 데 큰 도움을 줍니다. 몇 가지 추천할만한 규칙을 아래 치트 시트에 표시해보았습니다

⭐ <https://ko.javascript.info/coding-style>



JAVASCRIPT.INFO
The Modern JavaScript Tutorial

- AirBnB

airbnb/javascript

JavaScript Style Guide. Contribute to airbnb/javascript development by creating an account on GitHub.

🔗 <https://github.com/airbnb/javascript>



- NHN

코딩컨벤션

코딩 컨벤션은 읽고, 관리하기 쉬운 코드를 작성하기 위한 일종의 코딩 스타일 규약이다. 특히 자바스크립트는 다른 언어에 비해 유연한 문법구조(동적 타입, this 바인딩, 네이티브 객체 조작 가능)를 가지기 때문에 개발자 간 통일된 규약이 없다면 코드의 의도를 파악하거나 오류를 찾기 어렵다. 코딩 컨벤션 때문에 개발자 간 통일된 규약이 없다면 코드의 의도를 파악하거나 오류를 찾기 어렵다. 코딩 컨벤션은

🌐 https://ui.toast.com/fe-guide/ko_CODING-CONVENTION/



- 이 외에도 `JS coding convention`이라는 키워드로 검색하면 많은 자료를 보실 수 있어요.

▼ Python 코딩 컨벤션

- PEP8 - Python 공식 스타일 가이드

PEP 8 -- Style Guide for Python Code

The official home of the Python Programming Language

🐍 <https://www.python.org/dev/peps/pep-0008/>



- Google

google/styleguide

Table of Contents Python is the main dynamic language used at Google. This style guide is a list of dos and don'ts for Python programs. To help you format code correctly, we've created a settings file for Vim. For Emacs, the default settings should be fine.

🔗 <https://github.com/google/styleguide/blob/gh-pages/pyguide.md>



- `Python coding convention`이라는 키워드로 검색하면 더 많은 자료를 볼 수 있습니다.
- 😊 지금 단계에서는 처음부터 코딩 컨벤션을 다 내 코드에 적용하려는 부담을 내려놓으세요. 이런 것들이 있구나~ 하고 키워드만 기억해놓으셔도 충분합니다!

【 한 걸음 더]

- 앞으로 할 내 프로젝트에 수업시간에 배운 이름 짓기(naming) 컨벤션만 적용해보기

▼ 앞으로 개발자 커리어에 관심이 있는 사람은 아래 키워드도 한 번 확인해보세요.

- 키워드 : `linter`, `클린 코드(Clean Code)`

`linter`를 이용한 코딩스타일과 에러 체크하기

홈쇼핑처럼 6번째 상품인 까르보돈까스는 부먹과 찍먹을 선택해야 합니다. 코딩도 둘중에 하나를 선택해야 하는 경우가 많은데 협업을 위해 코딩 스타일을 설정하고 규칙에 어긋난 코드를 한땀한땀 수정했던 순간이 떠올라 `linter`에 대해 이야기합니다. 코딩스타일, 코딩 표준이라고도 불리는 코딩 컨벤션

 <https://subicura.com/2016/07/11/coding-convention.html>



Clean Code: Naming

Names are the smallest building block of code. Names are everywhere: Function, class, file, component, and container all have names. Names are also the basic building blocks of clean code. There are some hard and fast rules to a good name. A good name should

 <https://medium.com/better-programming/clean-code-naming-b90740cbbe12>



Clean code is simple and direct. Clean code reads like well-written prose.

— Grady Booch, author of *Object-Oriented Analysis and Design with Applications*

용어를 알아보자

▼ CRUD란?

CRUD(크루드, 씨알유디)라고 읽습니다. 마법 주문이름 같기도 하죠?

- 기본적인 데이터 처리 기능인 **Create, Read, Update, Delete**의 두문자를 따서 **CRUD**라고 합니다. 대부분의 소프트웨어는 이 기능을 가지고 있고요. UI가 갖추어야 할 기본 기능 단위로 CRUD를 묶어 이야기합니다.
- Create** (생성) : 예. 사용자가 게시글 쓰기
- Read** (읽기) : 예. 게시글 보기
- Update**(갱신, 업데이트) : 예. 게시글 수정
- Delete** (삭제) : 예. 게시글 삭제
 - 여러분들이 나중에 소프트웨어 기능을 개발할 때 CRUD가 내가 구현해야하는 기본 세트구나! 하고 생각하시면 좋겠죠?

한 걸음 더!

- API를 설계하는 방식 중 하나인 `RESTful API`에서는 HTTP method 와 CRUD 를 하나 하나씩 매핑해서 쓰기도 해요. 우리가 Read 기능은 GET 방식을 쓰자! 라고 한 것처럼요.

▼ 클라우드란?

- 그동안 우리는 클라이언트와 서버가 같은 컴퓨터에 있기 때문에 별 다른 설정없이 접근할 수 있었어요.
- 만약 다른 컴퓨터(클라이언트)에서도 내가 만든 프로그램이 실행되고 있는 서버에 접근할 수 있게 하려면 어떻게 해야할까요?
 - 모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 접근할 수 있도록 해야해요.
 - 그리고 서버가 요청에 언제나 응답할 수 있게 서버가 항상 켜져있고, 웹 서비스가 항상 실행되어 있어야하죠.
- 여러가지 설정을 해서 내 컴퓨터를 항상 켜두어도 되지만, 우리는 웹 서비스를 실행하기 편하도록 `클라우드`에서 서버 사용권을 구입할 거예요.
- ? 클라우드 많이 들어보긴 했는데 무슨 뜻인가요?
 - ! 클라우드(인터넷)을 통해 컴퓨터의 리소스를 사용할 수 있는 것입니다. 여기서 컴퓨터의 리소스는 컴퓨터를 이루고 있는 메모리, 저장장치(하드디스크, SSD), CPU 등을 이야기합니다.

- 네이버 클라우드, 구글 드라이브에 파일을 저장하고 읽어올 수 있죠? 바로 원격으로 쓸 수 있는 저장장치를 산 것과 같죠.
 - 우리는 AWS(Amazon Web Service)라는 곳의 EC2라는 서비스를 사용할 거예요. 원격으로 어떤가에 내가 사용할 수 있는 컴퓨터를 샀다고 생각하면 됩니다.
- ▼ [열 걸음 더 ] 클라우드 컴퓨팅의 특징 - 스kip하셔도 괜찮아요!
- 미국 국립표준연구소(NIST)가 정의한 클라우드 컴퓨팅은 아래와 같은 특징을 가지고 있어요. (원문. [The NIST Definition of Cloud Computing](#))

5 가지 주요 특징 Essential Characteristics

- On-demand self-service: consumer가 컴퓨팅 자원을 요구하는 즉시 자동으로 제공 (인간의 개입이 불필요).
- Broad network Access: 어디 있던 인터넷을 통해 리소스에 액세스
- Resource pooling: provider는 리소스 풀을 확보해서 multi-tenant model로 제공. 규모의 경제. 고객은 리소스 위치에 대해 신경쓸 필요없음.
- Rapid elasticity: 탄력적으로 리소스를 줄이거나 늘릴 수 있음(scale up and down). Capabilities can be elastically provisioned and released
- Measured service: 리소스 사용량이 측정되어서 쓴 만큼만 지불함. 투명성, 리소스 모니터링, 제어 및 보고 가능

3가지 서비스 모델 Service Models

- Software as a Service (SaaS) : 소프트웨어처럼 바로 사용할 수 있음. 예를 들면, MS 오피스 365, 구글 클라우드, 네이버 클라우드
- Platform as a Service (PaaS) : 응용 프로그램(Application)을 작성하고 실행할 수 있는 환경을 제공하는 것. 예를 들면, Google App Engine, Heroku
- Infrastructure as a Service (IaaS) : 사용할 수 있는 인프라를 제공하는 것. 예를 들면, 우리가 사용할 AWS, GCP

4가지 배포 모델 Deployment Models

- Private cloud : 비공개 클라우드. 단일 조직에서만 독점적으로 사용하는 것.
- Community cloud : 특정 용도로만 제한된 조직에서만 사용하는 것.
- Public cloud : 공개 클라우드. 일반인이 공개적으로 사용할 수 있는 것. 대부분의 클라우드 서비스는 여기에 속하겠죠?
- Hybrid cloud : 두 종류 이상의 클라우드로 구성된 것.

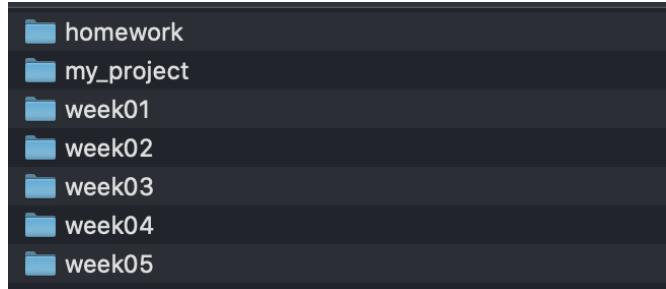
개발자 마인드 장착하기

▼ 개발자처럼 파일 관리하는 방법

- ! 개발자의 보물은 바로 코드! 내 보물이 어디있는지 모르면 난감하겠죠?
 - 1) 어떤 역할을 하는 폴더와 파일인지 한눈에 파악할 수 있게
 - 2) 다른 사람들과 협업할 때도 함께 정한 규칙대로 관리하면 유용하답니다.
- ▼ 1) 파일/폴더 이름 짓기(naming) 기본 규칙
- 폴더/파일이 어떤 내용인지 파악할 수 있게 적기
 - 개발자는 이름짓기(naming)을 정말 중요하게 생각한답니다. 특히 웹 프로그래밍은 데이터를 주고받는 과정입니다. 어떤 데이터(내용)을 담고 있는지 한 눈에 알 수 있는게 좋아요. 앞으로 배울 내용에도 꾸준히 이름짓기(naming)에 대한 내용이 나을 거예요.
 - 예) a → 무슨 폴더지? / homework → 숙제 폴더구나
 - 파일과 폴더 이름은 영어로 : 가끔 컴퓨터가 한글을 인식하지 못하는 경우가 있어요.
 - 특수문자는  만 사용하기 : 다른 특수문자(띄어쓰기 포함)을 컴퓨터가 알아듣게 하려면 조금 수고로워요. 우리는 단어를 연결할 때,  를 사용하겠습니다.

▼ 2) [👏튜터와 같이] 수업 폴더 만들기

- 8주동안 사용할 수업 폴더를 만들어 보겠습니다.
- 바탕화면에 sparta 폴더를 만들어주세요. 폴더 안에는 아래와 만들어주세요.



- week01, week02, week03, week04, week05 : 매 주차 수업에서 배운 파일을 저장할 폴더
- homework: 숙제 폴더
- my_project: 6~8주차 진행할 내 프로젝트

▼ 도구(프로그래밍 툴) 잘 사용하기

- 도구를 손처럼 사용한다면, 생산성이 올라간답니다! 하나의 도구를 강력하게 잘 사용해보세요. 도구 단축키와 다양한 기능들을요!
- 우리가 사용하는 Pycharm 과 크롬 개발자 도구를 잘 사용하기 위한 간단한 튜토리얼을 소개합니다.
- Pycharm 시작하기

리소스 - 문서 | PyCharm

리팩토링, 디버거, 코드 완성, 즉석 코드 분석, 코딩 생산성을 지원하는 지능적인 Python IDE

 PyCharm
The Python IDE
for Professional Developers
JetBrains

<https://www.jetbrains.com/ko-kr/pycharm/documentation/>

- 크롬 개발자 도구 사용하기

Chrome DevTools | Google Developers

Chrome DevTools는 Google Chrome에 내장되어있는 웹 저작 및 디버깅 도구셋입니다. DevTools를 이용하여 사이트를 반복하고, 디버깅하고, 프로파일링할 수 있습니다. 많은 DevTools 문서는 최신 Chrome 피쳐를 제공하는 Note: Chrome Canary 를 기반으로 작성했습니다. 완전히 반응하는 모바일

<https://developers.google.com/web/tools/chrome-devtools?hl=ko>



크롬 개발자 도구 101

썸머노트를 개발하며 늘 사용하는 크롬 개발자 도구에 대해 한 번도 제대로 공부해본 적이 없었는데, 마침 아살(@ahastudio)님이 "아듀 2018"이라는 일종의 Advent Calender를 주최하는 것을 보고 재미있겠다 싶어 이걸 기회 삼아 크롬 개발자 도구의 기초 내용을 정리해보게 되었다. 크롬 개발자 도구에 대한 정보

<https://lqezi.github.io/blog/chrome-dev-tool-101.html>

Chrome으로 디버깅하기

좀 더 복잡한 코드를 작성하기 전에, 디버깅이란 것에 대해 이야기해봅시다. 디버깅(debugging)은 스크립트 내 에러를 검출해 제거하는 일련의 과정을 의미합니다. 모던 브라우저와 호스트 환경 대부분은 개발자 도구 안에 UI 형태로 디버깅 툴을 구비해 놓습니다. 디버깅 툴을 사용하면 디버깅이 훨씬 쉬워지고, 실행

<https://ko.javascript.info/debugging-chrome>



JAVASCRIPT.INFO
The Modern JavaScript Tutorial

▼ 개발일지 - TIL(Today I Learned)

- ? TIL(Today I Learned) 이 뭔가요? 일일 개발일지 같은 건가요?

- ! 네, 맞습니다! 오늘 내가 배운 것(Today I Learned) 을 기록하는 거죠. 오늘 공부하면서 배운 걸 자유롭게 적는 것입니다. 기록하면서 내가 배운 내용이 한 번 더 정리가 되겠죠? 우리가 수업시간에 **소화타임**을 가지는 것과 비슷하죠.

개발자들은 Github repository에 적기도 하고 블로그를 사용하기도 합니다. 일일커밋(Git에 매일매일 commit과 함께 진행하기도 하죠.

- TIL은 아래처럼 토픽별로 적을 수도 있고

jbranchaud/til

Today I Learned A collection of concise write-ups on small things I learn day to day across a variety of languages and technologies. These are things that don't really warrant a full blog post. These are things I've picked up by Learning In Public™ and pairing with smart people at

⌚ <https://github.com/jbranchaud/til>



ohahohah/TIL

Today I learned :+1: / 자유롭게 배운 내용을 정리함.. Contribute to ohahohah/TIL development by creating an account on GitHub.

⌚ <https://github.com/ohahohah/TIL>



- 일자별로 적을 수도 있어요

6개월간의 TIL 회고 (꾸준히 하면 좋은 일이 생긴다)

2016년 퇴사 후 비전공자로 개발공부를 하면서 여려번 좌절하고 중간에 도망(?)을 가기도 했었다. github 커밋로그를 보면 학습 과정의 부침을 그대로 볼 수 있는데 아주 간단하게 정리하면 아래와 같다. 과거 : 일 문과 졸업 후 스타트업에서 여러가지 일을 맡아서 함 (마케팅, 정산, 고객 CS, 서비스 운영 등등) 2016년 9

⌚ <https://wayhome25.github.io/til/2017/08/14/TIL-for-6-months/>



- TIL하고 나서 한 달, 6개월, 1년 어땠는지 회고를 하기도 합니다.

TIL을 1년동안 진행하며

저는 개발자분들의 블로그나 NHN이나 네이버, 우아한형제들같은 회사 기술 블로그를 굉장히 좋아합니다. 개발 공부를 이제 막 시작한, 1년 전 어느날도 평소와 마찬가지로 여러 기술블로그들을 탐닉하는 와중.. 우연히 1일1커밋이라는 글을 봤습니다. 또 어디선가는 TIL이라는 글도 봤습니다. 이 TIL과 일일커밋에 관한 https://junwoo45.github.io/2019-09-10-til_후기/?fbclid=IwAR2GIPJ5NUzilp-NAtsvqfu1RcN2C9JrUFPWZ_itLqKvbjM98GQpC5yscSg

- **AngularJS-Atom** : Angular-UI 팀에서 메인테이너로 참여하는 Atom 패키지
- **atom** : Atom 에디터
- **electron-starter** : Electron(구 Atom Shell)
- **iots-ko** : iots 한국어 번역팀
- **FRENDS** : FRENDS 사이트
- **involved** : 개인 프로젝트
- **ansible-prototype** : 아는 사람들과 진행하는 사이드 프로젝트의 프로토타입
- **Socket.IO Examples** : 2년 동안 버려뒀던 개인 프로젝트

【📝 실행하기】

- 우리는 앞으로 '내 프로젝트'를 진행하면서 개발일지를 적어볼 거예요.
여기에 프로젝트 진행하면서 배운 것도 함께 적어볼까요?. 꼭 매일매일 적을 필요는 없어요. 공부할 때마다 조금씩이라도 적어나가면 된답니다.

▼ 가르치면서 배우기 Learning By Teaching

- 앞으로 스스로 공부를 위해 Learning By Teaching! 바로 가르치면서 배우기 학습방법을 소개하려고 합니다.
- 시험기간에 친구들을 가르쳐주면서 나도 자연스럽게 학습이 되었던 기억 한 번 쯤은 있으셨을거에요. 다른 사람에게 가르치면서 나만의 언어로 정리가 되고, 내가 모르는 부분을 깨닫게 되죠.
- ▼ 참고. Learning By Teaching 이 왜 효과적인지(교육 심리학 연구 요약 / 영문)

Learning by teaching others is extremely effective - a new study tested a key reason why

By Christian Jarrett The learning-by-teaching effect has been demonstrated in many studies. Students who spend time teaching what they've learned go on to show better understanding and knowledge retention than students who simply spend the same time re-studying. What remains unresolved, however,

⌚ <https://digest.bps.org.uk/2018/05/04/learning-by-teaching-others-is-extremely-effective-a-new-study-tested-a-key-reason-why/>



The learning benefits of teaching: A retrieval practice hypothesis

Teaching educational materials to others enhances the teacher's own learning of those to-be-taught materials, although the underlying mechanisms remain largely unknown. Here, we show that the learnin...

 <https://onlinelibrary.wiley.com/doi/abs/10.1002/acp.3410?campaign=wolearlyview>



- 특히 이 과정에서 **내가 무엇을 모르는지** 깨닫는 게 된답니다. 내용은 몰라도 괜찮아요. 원 모르는지 알게 되면, 이제 알아보면 되니까요 😊
 - 웹을 관통하는 개념! **웹 기초 동작 원리**를 20분 안에 무려 4번 복습할 수 있는 꿀팁을 알려드릴게요! 스터디에서도 응용할 수 있겠죠?
 1. 튜터님이 설명하는 웹 기초 동작원리를 열심히 듣는다.
 2. 빈 종이에 웹 동작원리를 스스로 그려본다.
 3. 옆 짹꿍 튜티에게 서로 웹 동작원리를 설명해준다.
 4. 칠판에 나와서 웹 동작원리를 그리면서 동료 튜티에게 설명해준다.
- [ 한 걸음 더]
- 앞서 배운 웹 동작원리 복습 꿀팁을 실제로 해본다!

단축키

- Notion : 모든 토글을 열고 닫는 단축키
 - Windows : alt + ctrl + t / Mac : option + command + t
- 크롬 개발자도구
 - Windows: F12 또는 alt + ctrl + i / Mac : option + command + i
 - 화면에서 "마우스 오른쪽 클릭 → 검사"도 가능!
 - 크롬 개발자 도구 사용 설명서 ([링크](#))
- Pycharm
 - html을 chrome으로 확인하기 단축키 : alt(또는 option)+f2 → shift 누르고 크롬 클릭
 - 파일 생성 : ctrl(또는 command) + n
 - 파일 실행 : Windows - ctrl + shift + F10 / Mac - ctrl+ shift + r
 - 더 많은 단축키는, Pycharm에서 확인하실 수 있습니다.

▼ 화면

