강준모
rapier83@gmail.com

공공 코딩교육 플랫폼

경제시뮬레이터만들기 그리고 세금의 실험

@ Huzen Dev. House

13 September 2017

1. 들어가면서

지난 8월

일산에서 나의 선생님이셨던 사회교사 A선생님을 찾아뵈었다. 그간 무슨 일을 하고 살았는지 어떤 일들이 있었는지 대화하다, 교직의 길을 가지 않고 회사원으로 살다보니 이제는 3년차 초보개발자가 되었다고 말씀드렸다. 그때 마침 "아 맞다!"고 손뼉을 치시더니, 한숨을 쉬며 '그 놈의' 융합교육이야기를 꺼내셨다. 여간 쉽지가 않은 모양이다. 그런데 융합교육에 꼭 코딩교육이 끼어있었다. 꼭 코딩이 아니어도 융합교육을 할 수 있을 텐데 왜 그런지 의아했다. 조금만 더 생각해보니 우리네 생각에 융합은 이과와 문과 과목간의 융합을 공식처럼 떠올렸다. 구글에서 융합교육을 검색하면 제일 처음 나오는 것이 한국과학창의재단의 '융합인재교육 STEAM' 이었다. 우리나라는 영국과 미국에서 쓰는 STEM에 예술Art을 덧붙였다. A빼고 STEM(Science, Technology, Engineering, Math) 은 모두 우리나라 기준으로 이과과목들이다.왜 문과 이과과 나누냐는 주제는 지금 논할 거리는 아니다. 다만, 문과에 있던 분들이 이과의 과목을 모르지는 않는데 다루기 어려워 하는 것 같다.

문과에 있다해서 이과의 과목을 모르는 것은 아니다. 반대의 경우도 마찬가지다. 특히, 사회과는 학부강의, 연수, 임용시험 등 전공관련 공부를 할 때 경제부문에서는 수학을 많이 이용한다. 사범대 학생때를 돌이켜보면 거시경제를 공부할 때는 수학이 반이라고도 했다. 아직 연락이 닿는 후배에게 물어보니 물론 지금도 그렇다고 한다. 수능시험에서도 경제과목 마지막 문제는 다분히 이과적이기도 하다. 수능이 아니더라도 적어도 함수의 개념은 알고 있어야

한다. 구분이 있다고 다른 쪽의 개념을 모르는 것이 아니다. 진짜 문제는 따로 있는 것 같다. 요즘 소프트웨어 인재 양성이 중요하다고 한다. 그러면서 자연스럽게 소프트웨어를 만드는 행위를 일컫는 코딩이 대두되었고, 너도나도 코딩을 해야한다, 코딩교육이 의무화 되어야 한다며 사람들이 하나둘 프로그래밍에 관심을 가지기 시작한 것이다. 프로그래밍은 과목으로 따지면 공학, 기술에 속하다 보니 이 분야를 공부하지 않으면 접하기 어렵다. 소프트웨어, 프로그램을 사용하는 것은 생활이 되었다. 글을 쓴다 그러면 당연히 워드프로세서를 열고, 무언가를 찾으려면 인터넷을 통해 검색한다. 종이와 펜을 드는 것은 다소 특별한 취미가 되었고 도서관은 컴퓨터가 가득한 공간으로 변신하는 중이다. 게다가 책상 위의 컴퓨터는 손바닥 위 스마트폰으로 옮겨왔다. 프로그램이라는 말은 이제 어플리케이션으로 바뀌고 앱이라고 줄여 쓴다. 여기서 진짜 문제가 드러난다 우리 생활을 윤택하게 만들어준 프로그램(앱)을 쓸 줄 아는 것에서 만드는 것으로 넘어가야 하는 것 아닌가?

* 이 장은 공공코딩교육플랫폼을 만들기에 대한 내용에 앞서 코딩 교육의 현실 진단과 비판에 대한 내용이므로, 이 상황을 대체로 아는 분들은 다음장으로 넘어가셔도 좋습니다.

골칫덩어리 융합교육

쉬운 사례에서 어려운 문제로

지구과학에서 지질연대와 지층에 대한 개념을 배우는 시간에 지리시간에 배운 우리나라 및 세계의 석탄 및 석유 분포를 통해 해당 지층이 어느 지질연대에 속했는지 파악할 수 있다. 이

정도면….수업 이전에 교과 선생님들 간의 간단한 토론을 통해 확인 할 수 있으며, 단순히 타 교과 교과서를 살펴 보는 것 만으로도 충분히 수업 설계 시 반영이 가능하다. 하지만 여기서 더 깊은 수준을 체험하기 위해서는 컴퓨터와 인터넷이 동원되어야 하고, 특별한 데이터를 다룰 줄 아는 기술이 필요해진다. 선생님들이 모르고 있는 것은 아니지만 학생들에게 어떻게 소개를 해야할지 부터 이어지는 문제는 정말 어렵다.

지구과학, 지리간 학습을 마치고 석유를 비롯한 지하자원의 가격 변동과 물동량에 대해서 알아본다고 해보자. 아마 교과서에는 세계지도위에 석유 수출을 화살표로 표현한 그래프가 나와 있을 것 같다. 이를 최신 데이터를 이용해 인포그래픽스로 만들어 보는 미술, 사회, 수학의 융합수업을 상상해보자. 세계은행, 국제석유기구 등을 통해서 자원의 수출입 데이터를 구한다. 각 국가별 수출량을 추려서 0 ~ 1 까지의 상대값을 구한다. 이를 화살표의 크기에 곱해서 지도에 방향을 맞춰 그려넣는다. 보다 정확히 그려 넣으려면 길이를 피타고라스 정리를 이용해 좌표값으로 옮긴다.

이를 다시 쉽게 생각해보자. 주제는 세계지하자원 수출입 동향 그래프다. 교과서는 물론이고 신문에서도 자주 볼 수 있다. 그러나 만들려고 하면 여간 골치아픈 것이 아니다. 그냥 선생님이 미리 추려놓은 간단한 표를 주고 나누어준 포트폴리오 자료에 펜으로 직접 눈대중으로 그려보는 수준으로 끝낼 수도 있다. 그러나 이런 수준의 체험은 학생들이 사회 진출 한 후에 별 쓸모가 없다. 만약 사회에 진출한 뒤 이와 같은 일을 하여야 한다면, 어려운 문제에 해당하는 작업들 중 하나를 해야하기 때문이다. 데이터를 정리해오거나, 인포그래픽스로 표현을 하거나. 여기서 협업에 관한 이메일, 클라우드 문서 공유 같은 작업은 모두 제외했다.

요즘과 현장

이 융합 수업설계의 경우 긴 시간의 연구가 필요하며 시행을 위해 교육과정의 재구성 및 필요한 것이 이만 저만이 아니다. 교과서에서 다루는 것 이외의 내용을 가르쳐야 하기 때문이다. 데이터를 추려서 2차 데이터로 가공하는 과정에서는 엑셀과 같은 스프레드 시트를 이용해야 하고, 인포그래픽스로 표현하려면 알맞은 세계지도를 검색포털을 통해 찾은 뒤 파워포인트, 포토샵과 같은 프로그램을 다룰 줄 알아야한다. 예전에도 그랬지만, 이런 과정에서 꼭 반에 한명씩 선생님도 부러워하는 '도사'가 있지 않았었나? 그리고 학교마다 그런 '도사'들 중에서 소위 전국구로 이름 날리는 특기생도 종종 있다.

그 많은 엑셀개발자는 어디로 갔을까?

전문가의 영역에서 순식간에 벗어난 프로그램: 엑셀 스프레드시트

과거 윈도우 95가 나오기 이전의 스프레드시트(엑셀 등)는 전문가들만의 영역에 속했다. MS 오피스가 출시 되고 업무에서 유용한 것이 하나씩 증명되면서 워드, 엑셀, PPT 이 세가지는 어느새 필수 역량이 되었다. 파워포인트는 쉬운 편에 속하니 논외로 하자. 다만, 전문가의 도구였던 스프레드시트는 99년 컴퓨터활용능력시험이 시작되면서 크게 대중화 되었다. 미국 등 영미권에서는 아직도 엑셀개발자, 엑셀프로그래머, 엑셀엔지니어라는 직업이 있다. 쉽게 짐작할 수 있듯 주어진 문제를 엑셀로 해결하고 엑셀로 표현해 내는 직업이다. 우리나라에서는 엑셀개발자를 따로 두지 않는다. 물론, 엑셀에서 제공하는 프로그래밍 언어인 VBA (Visual Basic for Applications) 까지 다루는 사람을 엑셀 개발자로 부르기는 하지만 아주 전문적인 영역에서 엑셀을 다루어도 주류 프로그래밍 언어인 VBA 를 이용하지 않고도 엑셀에서 기본적으로 제공하는 함수를 이용해 계산해 낼 수 있다. 이를테면 두 데이터의 상관계수를 구해야하는 문제가 있다면 `=CORREL(B3:C14)` 을 입력하면 나온다.

한가지 사례를 더 들어보고자 한다. 거의 대부분의 사회과학 분야 학자들은 SPSS나 SAS같은 통계 전문 프로그램을 쓴다. 대학원생이 아니더라도 대학생을 비롯해 학문의 근처에 있었던 사람들 사용하지는 않아도 통계 자료로 가설을 입증해 내려면 이런 것들을 써야한다는 것쯤은 알게된다. SPSS나 SAS는 사용해본 사람들은 느끼는 것이 엑셀의 연장선이라고들 한다. 간혹 두 프로그램이 워낙 고가여서 개인적으로 통계 자료를 처리하려고 할 때에는 엑셀의 심화된 도구를 사용한다. 요즘에는 구글 스프레드시트, 클라우드협업 문서 등을 이용해서

한번에 여러사람들이 한 프로젝트에 붙어 한쪽에서는 카카오톡 등으로 채팅하며 작업을 순식간에 끝내기도 한다.

그리고 요즘들어 자주 보였던 특이한 한가지. 알파고와 이세돌이 대국을 한 이후로 빅데이터 교육이 필요하다며, 대학생과 직장인을 대상으로 공개 통계 프로그램인 `R` 교육이 유행한 적이 있다. R프로그래머의 경우 수년전만해도 우리나라에서는 찾아보기 힘들 정도로 드물었고, 혹시라도 전문가가 있다면 평생 먹고살 걱정은 없다고들 했다. 빅데이터 시대에 발맞추기 위해서는 DB 언어인 SQL등을 배워야 할텐데, 왜 이런 비주류 프로그램이 왜 난데없이 유행을 탔을까? 지금은 다소 사그라 들은 것 같다. 그렇다고 R전문가를 많이 배출해 낸 것 같지도 않다.

10년전 공대, 10년후 경영대

R은 프로그래밍 언어에서 고급언어로 분류한다.[1] 기계보다는 사람이 알아보기 쉬운 언어이다. 보통 프로그래밍을 한다고 할 때 현업에서는 JAVA와 C/C++ 라는 언어를 사용하는데, R은 C 언어를 통계에 맞춰 개량한 프로그래밍 언어이다. 10년전에는 통계학과에서나 간혹 배우고, 연구 분야에서는 SPSS나 SAS로도 감당이 안되는 대용량을 실험데이터를 다루게 되거나, 셀기반의 방식이 불편하거나 표현에 한계가 있는 경우에 사용하는 언어다. 그런데 이 비주류 언어를 경영대에서 가르친다는 소식을 자주 보였고, 여러 사설 교육기관에서도 특별과정을 개설했다.

10년전 마케팅, 브랜딩을 공부하는 것이 유행이었다. 학교마다 다르겠지만 산업공학과나 산

업공학전공이 있는 과 등은 전공과목에 경영학, 브랜드이론 과목이 있었다. 이런 추세에 따라서 기술경영, 정보경영과 같은 융합과목들이 생겨났다. 공과대학말고도 언론정보학, 신문방송학, 광고홍보학과 때로는 시각디자인학과 같은 학과 학생들도 자연스레 경영학, 소비자행동론을 배워 경영학과에서 석박사 학위를 받는 경우도 있었다. 국문과나 문예창작과는 콘텐츠 경영을, 정책학 무역학과에서는 국제 경영학, 글로벌 마케팅 등을 배웠다. 서울대에서 기술경영과목은 공학대학에서 과정을 두고 있기 때문에 이를 전공하기 위해서 경영학과 생들이 공과대학으로 넘어가서 공학박사 학위를 받는다. KAIST에도 테크노경영대학원이 있어 많은 경영학도들이 공대에서 학위를 받았다.

10년전 공대에서 경영학을 가르치더니 10년후 지금 경영대에서 프로그래밍을 가르친다. 이를 고등교육의 발전으로 봐야하나 유행에 따라 움지이는 것으로 봐야하나. 만약 후자라면 고등교육부문도 크게 걱정된다.

심지어 코딩(프로그래밍)교육이라니?

일반 사무 분야의 코딩교육

일반 사무 분야에서도 코딩 교육의 바람이 불고 있다. 삼성에서는 SW인재 육성을 위해 엄청난 돈을 투자하기로 결정하기도 했다. 이 외에 온라인 강좌를 제공하는 이러닝서비스가 빠르게 성장했다. 이들의 고객은 취업준비생이 주류를 이루고 현재 재직중인 사람, 이직을 준비하는 사람들도 많이 이용한다고 알려졌다. 특히 사무직에서 Data Scientist등 데이터 비즈니스 분야로 전직하기 위한 경우도 있다.

R의 사례

앞서 R에 대한 이야기를 한 것은 일반 사무에서는 엑셀로 하는 일을 R로 충분히 해낼 수 있으며 다루는 DB의 크기와 속도의 차이도 엄청나게 다르기 때문이다. 또한 R 전문가들이 꾸준히 프로그램을 발전시켜 사무업무에 바로 활용할 수 있도록 그래프, 지도가 있는 그래프, 표가 들어간 보고서를 우리가 쉽게 접하는 MS워드, PDF 등으로 쉽게 만들어 내는 도구들도 개발해 냈다. 개인이 이를 활용해 공격적으로 학습에 나서면 업무 효율이 크게 올라가기도 한다. 전산업무인력에서는 알고리즘에 대한 공부가 크게 증가했다고 하고 시험등으로 평가를 하는 경우도 있다. 비즈니스 세미나의 경우 프로그래머가 아닌 사람이 업무 도구를 만들어 효율성을 제고한 사례를 보고하는 자리도 있다. 물론 R 뿐만이 아니라 우리가 다루고자 하는 Python 부터 프로그래밍 기본 언어인 C 까지 비개발 조직에서도 프로그래밍이 다양하게 활용되고 있다.

코딩 의무 교육: 해외사례

교육에서 해외사례를 벤치마킹하면 코딩 교육에서 과학, 공학 교육으로 코딩교육을 바라보며, 여러 회사들이 교육을 위해 제품이나 언어를 만든 것을 확인해 볼 수 있었다. 그중 특히 영국이 눈에 띈다. 초등학생 수준은 코딩 교육에서 더 나아가 하드웨어까지 발전했다. Raspberry PI 와 같이 속이 다 드러난 카드만한 컴퓨터에 각종 전자부품이나 알루미늄 호일 등을 이용한 전자 드럼이나 나만의 문열림 알람기같은 조잡하지만 그럴싸한 전자장난감을 만들기도 한다. 가격은 약 5만원 선이다. 과거 방학때 <탐구생활>에 있던 건전지와 자석으로 간이 모터를 만들었던 것과 비슷해 보인다. Raspberry PI 를 만드는 회사는 Raspberry 재단으로 자사 제품을 활용하여 어떤 교육이 이루어지고 있는지, 또는 어떻게 쓸 수 있는지 사례 영상을 제공해 학교 선생님들을 위한 코너도 마련해두었다.

미국의 경우, MIT에서 초등교육용 프로그래밍 언어인 Sketch를 개발했다. 명령어를 치면서 코딩을 하는 것이 아니라 명령어 블록들을 조합하면서 프로그램을 만든다. 이런 방식은 기계 플랜트와 같은 전문 분야에서 활용하던 방식을 학생들이 쓸 수 있게 개조한 것이다. 앞서 소개한 Raspberry PI와 호환이 잘 되며, Sketch로 프로그래밍한 생일축하 웹페이지 등을 만들었던 교육 사례를 소개한다.

LEGO사의 경우 Mindstorm EV3 라는 세트를 5,6 학년 수준에서 부터 대학생 엔지니어링 교육에 활용 할 수 있게 만들었다. 해당 셋트로 어떤 주제로 교육에 활용이 가능한지 보여주

는 사례들이 있다. 예를 들어 빛의 3원색을 가르치는데 이 제품에 포함된 컬러 센서를 이용해 빛의 3원색 마다 블럭의 색에 반응하는 값을 확인하고 3원색이 합쳐지면 흰색으로 판정하는 프로그램을 볼 수 있다. EV3는 완구용 세트, 교육용세트로 따로 구분하여 판매하여, 학교 등 교육기관의 구매를 유도한다.

이 움직임에서 주의 깊게 살펴 볼 것은 이 사업에 핵심은 수익 극대화가 아니고 교육 효과의 극대화다. 특히 MIT, Raspberry 재단은 비영리 기관으로 기부 또는 제품 생산에 필요한 비용만 받고 있다. LEGO역시 해당상품의 교육용 버젼은 브랜드의 공격적인 마케팅 보다는 엔지니어링 교육 사례를 소개하며 수요를 만드는 정도다. 공공코딩교육플랫폼이 추구하는 것과 같다.

코딩 의무의 속내

코딩 의무교육에 대해서 현장 엔지니어(개발자, 프로그래머)들은 어의도 없고 어처구니도 없다는 반응을 내놓았다.[2] 다양한 프로그래밍을 초중교육에 배우는 것은 한계가 있는데 그 짧은시간에 주요과목도 아닌 이 기술을 얼마나 가르치겠냐는 의구심 때문이었다. 학부모는 오히려 관심있는 학생들만 모아 동아리 활동으로 할 수 있게 만드는 것이 좋다고 했다.

학생들을 가르치는 선생님들도 많이 힘들어 한다. 특히 전산업무를 많이 했던 교사들은 도데체 어떻게 시작을 해야할지 모르겠다는 반응이다. 현업계나 교육계나 코딩 교육의 필요성, 추상적인 기대효과, 목표들만 서술이 되어있지 정작 당장에 가르칠 환경도 방법도 없다. 상세히 말하면 산업현장에서 쓰이는 C 나 JAVA를 가르칠 것인지, 최근에 나온 고급언어인

Python, Ruby 등을 가르칠 것인지, 그것도 아니면 학습용 프로그래밍 언어인 Sketch 에서 출발해 하나하나 학습해 나갈 것인지 정해진바가 없다. 당장의 현실을 돌파하려면 개인이 업무나 방과후를 활용해 공부하는 방법 외에는 없는 것 같다.

이 사례들에는 공통으로 "나중에 사회에 진출했을때…" 가 담겨있었다. 대체로 코딩 교육을 프로그래밍적 사고를 하는 것 보다 미래에 취업 스펙 정도로 생각하는 것으로 보인다.

우리는 이를 심각한 문제로 진단한다. 애초에 코딩교육도 세계적으로 컴퓨터 교육이 시작되고 있기 때문에, 경쟁력 강화를 위한 목적으로 도입되었다. 코딩 교육은 현재 우리나라에서 사용하고 있는 교육과정의 일환으로 각 과목마다 응용해 볼 수 있다. 연산부터 검색, 모델링 등은 약간의 사전 교육만 주어진다면 충분히 수업설계가 가능하다고 본다. 즉, 코딩을 따로 교육하는 것이 아니라 학교 교육과정 전반에 컴퓨터를 활용한 수업이 이루어져야 한다. 그 이유는 간단하다. 우리 세상의 다양한 부분이 프로그램에 의해 이루어지기 때문이다.

인터넷 시대다, Web 2.0 시대다, 모바일 시대다 라고 하지만 결국 컴퓨터(더 첨가하면 컴퓨터가 담긴 전자기기)와 네트워크를 이용하는 것 자체는 변하지 않았다. 그간 사용하는 것에 집중했고 만드는 것에 대해서는 무관심 했기 때문이다. 우리는 이 무관심을 관심으로 돌려놓기 위해 공공 코딩 교육 플랫폼을 구축하고자 한다.

대안와 유사서비스

소프트웨어 기업들의 친절함.1: 문서화

세계 유수의 소프트웨어 기업들은 자사 소프트웨어를 출시 할 때 친절하고 자세한 사용법을 제공한다. 보통 Documentation & Tutorial 의 형식으로 제공한다. Documentation 은 사용방법에 대한 자세한 설명서이다. 제공하는 기능은 어디까지이고 어떤 매개변수(데이터)를 가공해 낼 수 있는지 설명한다. 때에 따라서 근간이 되는 수학 풀이도 제공한다. 설명서는 대부분 굉장히 방대하기 때문에 처음 사용하는 사람들을 위해 Tutorial(따라하기)를 제공한다. 최초에 셋팅은 어떻게 하고 어느 분야에서 유용하게 쓸 수 있는지 설명서에 어느 부분에 실려있는 부분을 활용하고 있는지 보여준다.

1) 예시 - Mongo DB

위 회사는 빅데이터 DB 를 다루는 소프트웨어를 만든다. Mongo University 라는 온라인 대학(학위는 없음)을 운영하며 XX101 과 같이 한 파트를 과목으로 만들어 영상을 보고 문제를 풀어보고 기한내에 과제를 제출해야 이수증을 받을 수 있다. 수준은 DB 를 처음 시작하는 사람부터 DB 전문가가 자사 소프트웨어로 옮겨 올 수 있게 꽤 높은 수준의 내용까지 담고 있다.

2) 예시 - Unity

3D 게임을 만들 수 있는 위 프로그램도 처음 사용해보는 사람도 따라하면서 하나의 완성된

간단한 게임을 만드는 것 부터, 게임 제작에 이용할 수 있는 Asset(사전에 제작된 3D 모델파일들) 스토어를 열었다. 이런 방식은 Unity 외 다른 서비스들도 많이 하고 있지만, Unity는 무료로 이용할 수 있는 폭을 넓게 잡아 많은 게임프로그래머들이 제작을 할 수 있게 도와줬다.

소프트웨어 기업들의 친절함.2:무료정책

일부 유명 소프트웨어는 무료정책을 가지고 있다. 3D 게임을 프로그래밍할 수 있는 Unreal Engine, Unity 은 기본적으로 무료이며, 수익이 나는 조직에게 사용료를 부과하게 한다. 물론 국내에서도 개인은 무료로 회사는 비용을 내고 써야하는 소프트웨어들이 있다. 하지만 이들의 차별화점은 테스트 기간이 길고 수정이 많은 게임제작을 고려하여 완성된 작품이 나오기 전까지 무료로 쓸 수 있게 하거나 비용을 지불하는 경우 파격적인 서비스를 제공한다.

온라인 쇼핑몰로 유명한 Amazon은 자사 서버 및 클라우드 자원을 제공하는 Amazon Web Service(AWS)도 운영하고 있다. 사용자들이 제대로 경험해 볼 수 있도록 1년간의 Free-Tier 기간을 준다. 대용량 서비스의 경우 가격을 지불해야한다. 이 정책으로 AWS를 비롯한 타 서비스들도 1년 무료 사용 정책을 관행처럼 운영하고 있다. 만약 웹서비스를 만들어 보는 독립 개발자나 소규모 벤처, 스타트업의 경우 이 정책을 활용하면 최소 월 3만원 이상 지출되는 서버비용을 절감할 수 있는 수준이다.

공공 코딩교육 플랫폼이 꼭 필요할까?

국제적으로 본다면 무료 코딩 교육 서비스는 차고 넘친다. 그래서 이런 서비스의 전문 번역 팀을 구성하여 번역의 허락을 구하거나 번역의 공간이 있는 곳에는 집중하여 번역하는 작업을 고려하기도 했었다. 그러나 결국 어느 곳에 가야 적절한 교육을 받을 수 있는지 안내할 중심(Hub)이 필요하다. 그래서 그 중심을 잡는 역할과 더불어 학교 교육에서도 선생님과 학생이 함께 이용해 볼 수 있는 교육 프로그램도 만들어 보는, 우리나라의 사정에 적절한 플랫폼을 내놓기로 결정했다. 가장 큰 목적은 기술의 습득 이전에 기술의 변화를 보는 눈을 키워주는 것이기 때문이다.

2. 이렇게 한번 해봅시다

엑셀 배우듯이

수행평가와 대학 리포트

99년 수행평가가 도입 될 때 때 학교에서 보고서를 쓰게하는 경우가 있었다. 조별 발표수업을 하는 경우도 있었다. 대학에서 주로 하는 방식이 아닌가? 한가지 특이한 점은 대부분의 학생들이 한글 워드프로세서 정도는 쓸 줄 알았다. 파워포인트도 따로 교육이 없었지만 어느 다들 곧잘 다루었다. 물론 학습이 필요한 학생들도 있었겠지만 오랜 시간이 걸리지는 않았다. 반마다 컴퓨터 좀 한다는 친구한테 물어보면 된다. 엑셀을 배워야한다는 것에 대한 부담은 없었다. 오히려 부담으로 작용한 것은 취업준비생들이 자격증을 따는데 있었지만 엑셀을 배우는 것 자체가 부담이 되지는 않았다. 과거에 워드프로세서가 그런 것과 같아보인다.

막상 코딩을 해보면…

단순하게 이 상황을 풀어나가기 위해서 코딩의 기본이 무엇인지 간단히 짚어보자. 거의 대부분의 언어가 하나같이 비슷한 문법을 가지고 있다. 다만 기호가 조금씩 다를 뿐이다. 그리고 학습하는 순서는 대체로 다음과 같다.

1) 변수의 할당(a,b라는 변수에 각각 3,4의 값 저장하고, a + b의 결과는 7이다.)
2) 연산자 (사칙연산, 논리연산)

3) 언어별로 기본으로 제공하는 함수

4) 문자열 다루기 (출력, 자르기, 변형하기 등)

5) 기본 자료구조 다루기(수열과 같은 순서형 자료, 사전과 같은 키워드형 자료 등)

6) 제어문과 반복문

7) 사용자 정의 함수 (기능을 일괄로 수행하는 함수 정의하기)

8) 파일 입출력(결과를 txt 파일 등으로 저장하기)

9) 유용한 외부 기능 다루기 (자료를 엑셀파일로 저장하는 기능 등)

여기에 추가로 객체 및 객체 지향 문법등을 다루기도 하지만 이 역시도 큰 차이는 없다. 대부분의 입문서도 배우는 순서는 조사를 하지 않아도 이 순서에서 크게 벗어나지 않는다. 대학교재나 기술자용으로 나오는 문서들도 이런식으로 전개된다. 1)변수의 할당 ~ 4)문자열 다루기 그리고 6)에서 제어문 같은 경우에는 엑셀과 다른 점이 거의 없다. 엑셀에서 A1 셀의 값과 A2의 값이 같으면 "GOOD", 다르면 "BAD"라고 출력하는 함수를 A3작성한다고 하면 다음과 같은 코드를 입력한다.

```
      1     2     3
A1 | ____|_____|_____
A2 | ____|_____|_____
A3 | =IF(A1=B2,'GOOD','BAD')
```

이를 앞으로 중점을 두고 다룰 Python 이라는 언어에서 구현한다면 다음과 같다.[3]

```
A = input("첫번째 숫자를 입력하세요.")
B = input("두번째 숫자를 입력하세요.")
if A is B:
    print("GOOD")
else:
    print("BAD")
```

프로그램을 모르는 사람에게도 if 로 시작하는 제어문 부분은 오히려 Python이 자연언어에 더 가깝다. 중요한 것은 엑셀이나 Python 이나 코드를 구현하는데 큰 차이가 없다. 엑셀은 A=... B=... 대신에 한 셀에 값을 입력하는 것만 다르다. 그리고 엑셀에서 전문가 용으로 제공하는 VBA 언어의 문법은 Python과 거의 일치하다고 할 수 있을 정도로 비슷하다. 코딩은 우리에게 전혀 낯설은 분야가 아니다. 이렇게 엑셀을 배우면서 하나의 견적서나 보고서를 만들어 내듯, 코드를 하나하나 쌓아가면 하나의 프로그램을 만들어 낼 수 있다. 그리고 위에서 제시한 순서에 따라 언어를 익히고 몇가지 패키지를 활용하면 프로그램을 웹브라우저에 띄우거나 모바일 화면에 보일 수 있게 만들 수도 있다.[4]

코딩을 생활에 담그자

코딩은 이정도만 해도 벌써 기초의 절반을 끝냈다. 우리가 지난 2십여년간 생활에 엑셀과 PPT 등을 담궜듯이 각자 편한 프로그래밍 언어를 생활에 담근다면, 코딩교육도 한결 편하게

만들어 갈 수 있을 것 같다. 아니 오히려 자연스럽게 일어나는 현상이 될 것 같다. 그러나 이 현상을 두고 급박하게 제도의 복잡한 체계나 목표 그리고 한참 이후에 있을 기대효과를 벌써 세워 놓는 것은 되려 무의미해 보인다. 효용도 각 개인에게는 거의 없을 것 같다. 하지만, 우리나라에서 반 농담, 반 진심으로 이런 계획을 통해 수능 문제에 나온다거나 특목고입시, 대학입시에 반영 된다면 우리나라의 교육계가 우려하는대로 사교육이 범람하고 본래의 목적을 상실하게 될 것이다. 우리는 그 것만은 피하고자 한다.

해결의 실마리

Back to the Basic

현장의 대표적 언어는 C++와 JAVA, JavaScript (둘은 완전히 다르다). C++과 JAVA 는 각각 포인터, 객체 등에서 명령어가 너무 복잡하고 JavaScipt는 웹페이지 안에서만 쓸 수 있다는 점에서 한계. (물론 다른 방법을 이용하면 여러가지로 활용 가능하나 생략) 그런데 왜 이런 언어만 쓰는지 90년대 컴퓨터 학원이 미술학원과 비슷한 곳에 있던 시절 컴퓨터 학원을 다녀본 친구들은 대부분 GW BASIC 으로 시작했을 것이다.

BASIC은 당시 주요 언어였던 Fortran, COBOL, Turbo C 등을 익히기 전에 감을 잡기 위한 교육용 언어로 개발되었다. 키보드로 명령을 10, 20, 30, 단위로 치면 되었고 결과는 run 명령을 치면 바로바로 실행되었다. 어려운 사용자 지정함수나 객체는 아예 빠져있지만 BASIC 을 배우고 나면 다음 개념을 쉽게 배울 수 있다. 자주 쓰는 명령어들은 아래에 딱 10가지만 있어서 그것만으로도 "*" 로 크리스마스 트리 만들기도 했었다. 다시 그런 수준에서 부터 시작해보자.

하나 고릅시다.

그렇게 해서 나온 언어들이 꽤 많다. 앞서 잠시 언급한 바와 같이 우리는 Python 을 이용하려고 한다. 세상이 좋아져서 그런지 과거의 BASIC 과는 달리 이 단순한 언어만으로도 많은 일을 할 수 있다. 전문기술서도 Python 으로 알고리즘 이론, 대규모 빅데이터 처리를 다룬

다. 산업계에서는 NASA같은 연구기관에서 주요 언어로 쓰고, 코딩 스탠다드를 엄격하게 지키기로 유명한 Google은 C++, JAVA (이하 Native Language)에 Python이 공식언어로 등록되어 있다.[5] 산업계에서 쓰는 Native Language에 비하면 성능은 꽤 많이 떨어지지만 학습시간, 생산성(코딩하는 시간)이 월등이 좋다. 언어 처리의 속도가 떨어진다해도 컴퓨터가 워낙 빨라져서 코딩 교육의 수준에서는 문제가 되지않는다.

인터넷 중심

한가지 큰 문제가 있다면 어떤 언어를 개발하기 위해서는 개발환경을 만들어야 한다. Python이든 무엇이든 코드를 다 작성한다해도 이를 해석해서 컴퓨터에게 알려줄 도구가 필요하다. 그리고 변수나 명령어들이 많아지게 되면 이를 관리해주는 기능이 탑재된 에디터가 없으면 다소 복잡한 코드는 사소한 실수 때문에 시간을 많이 허비하게 한다. 이 도구들을 보통 에디터 또는 IDE라고 하는데 이를 고르는 것에도 시간이 꽤 걸린다. IDE마다 또 개발환경이 다르기 때문에 코딩-평가 에 집중할 수 없다.[6] IDE와 개발환경은 인터넷 환경에 구축하여 웹브라우져를 통해 코딩을 하게 할 예정이다. 실은 많은 튜토리얼, 교육 서비스들이 웹사이트에서 바로 코드를 편하게 작성하고 바로 평가 할 수 있게 해준다.

간단간단하게

처음부터 생활의 문제를 풀기 위해서 긴 코드를 작성하게 할 필요는 없다. 작은 결과를 하나

씩 구현하게 하고 그 결과들을 저장하면서 어느 정도 수준이 되면 실생활에 쓰일만한 프로그램이나 역사적으로 컴퓨터 산업 발전에 지대한 영향을 미친 알고리즘을 따라해보게 한다. 역사적 알고리즘은 대체로 간단하다. 보안 교육과 병행을 위해 암호화, 바이러스-백신 코드[7] 등을 직접 만들어 보고 해결방법도 찾는 방법이 있다. 자세한 구현 방법들은 성공사례에서 언급되는 학습 서비스를 체험해보는 것이 좋다.

성공사례

Khan Academy

Salman Khan 이라는 사람이 조카들의 수학교육을 위해 태블릿으로 수식을 하나씩 그리면서 녹음을 한 영상을 유뷰브에 올린 것이 교육 플랫폼으로 발전했다. 반응은 뜨거워져서 1+1이 몇이니? 라는 질문에서 상미분방정식에 이르기 까지 영상을 올렸고 이를 뛰어넘어 컴퓨터를 비롯해 역사, 사회문화, 경제 등 다양한 학문으로 퍼져나갔다. 수학은 학교 교육에 해당하는 부분들은 연습문제를 풀어볼 수 있게 연습장, 때로는 계산기[8]도 제공을 한다. 차라리 후원을 받고 Huzen Dev. House가 Khan Academy의 자료를 번역하자는 의견도 있을 정도였다.

Udacity

현재는 유료 강의가 대부분이지만 오픈 이전 테스트 때에는 모든 수준 높은 강의가 무료로 되어있어서 많은 개발자들이 도움을 받았다. 특히 Introduction of Computer Science 과목은 미국 버지니아대 David Evans교수가 Python 을 사용하는 법을 기초부터 하나하나 가르치면서 구글의 페이지랭크 검색 알고리즘을 소개한다. Huzen Dev. House는 이 모델을 유사하게 활용할 예정이다.

유다시티의 특징은 짧은 설명과 간단한 실습이다. 구글 검색 알고리즘은 Markov Chain 이라는 사전 지식이 필요하지만, 복잡한 것을 모두 제하고 사칙연산과 페이지 랭크 개념만 간

략히 설명하며, 간략히 구현해 볼 수 있게 도와준다. Georgia Tech과 정규 석사과정을 가장 처음 실시했다. 비용은 $7,000 (on Campus $45,000)현재 Nano degree 과정을 다수 개설해 웹사이트 개발에서 무인자동차 개발까지 현장에서 바로 쓸 수 있는 과정들도 개설해놓았다.

우리가 만들 공공 코딩교육 플랫폼은 Khan Academy 와 Udacity 와 유사한 형태로 제작될 예정이다.

COURSERA: Stanford Univ.

Andrew Ng(스탠포드)교수가 Machine Learning 강좌를 공개하면서 유명해졌다. 그는 빅데이터, 머신러닝 의 대중화에 기여한 가장 중요한 사람으로 뽑히기도 한다. 역시 통계학에서 중요한 개념들 중에서 수식이나 복잡한 부분들에 대해서는 간략히 설명하고 기계학습을 시행해 보는데에 중점을 둔다. 과제는 주어진 문서를 읽고 올바른 식을 입력하는 것이다. 수식이 제대로 들어가면 자동으로 그래프가 그려져서 스스로 다양한 테스트도 할 수 있게 한다. 많은 Data Scientist[9]들도 입문단계에서는 이 강의를 활용했다는 증언들도 많고, 몇몇 책에서는 이 강좌를 한번 들어보라는 권유를 하기도 한다.

COURSERA: Johns Hopkins Univ.

의료통계과 교수 3인이 Data Science Specialist 과정을 개설했다. 기계학습, 통계 실무 프로그래밍을 하는데 있어 필요한 도구들을 사용하는 법 부터 통계 이론 실제 응용프로그램 제작 및 보고서 또는 논문 조판까지 알려준다. 수강은 무료고 자격증(이수증)을 받을 때만 강좌당 $4.99를 지불해야하고 총 10강좌에 실습으로 Capstone Project 가 하나 있다. (약 $500 소요) 학위로 인정이 되지는 않는다. 어느정도 컴퓨터 지식을 가지고 있으면 충분히 도전해 볼만 하다.

3. 어떻게 만들 것인가?

기본 구성

공공

공공을 가장 먼저 둔 것은 누구나, 제약없이 이용할 수 있게 만들고 싶기 때문이다. 혹여라도 교육에 있어 학생으로 국한해버리면, 학생이 아닌데도 배우고자 하는 사람들에게는 장벽이 될 것 같기 때문이다. 코딩을 배우고 싶은 사람은 학생이 아닌 경우는 일반인 부터 학교에 다니지 않는 청소년, 활동에 제약이 있는 사람(제소자 등)이 있다. 이들도 아무런 장벽없이 이용하게 만들고자 한다.

웹서비스의 경우 초기 구축이 끝나면 서버 유지비, 웹사이트 안정화 등의 비용이 발생한다. 이를 감당하기 위한 방법은 사업모델 부분에서 설명하고자 한다. 이 초기 단계의 준비가 끝나면 개선(리뉴얼, 업데이트 등)을 제외하고는 크게 비용이나 자본이 필요치 않게 된다. 소모되는 것도 아니라서 공공재의 비극이 발생할 일이 없다. 다른 사업보다 관리, 유지 비용이 훨씬 적게 든다.

혹여나 사용자수가 많아져 서버를 증설해야하는 경우는 지금 고민할 거리가 아닌 것으로 판단된다. 다국어 서비스를 하지 않는 이상 사용자의 상한은 충분히 예측 가능하다. 그리고 대응하기 위한 사용자 수용 용량(Capacity)은 그리 크지 않을 것으로 예상한다. 사용자 수용 용량의 문제는 그 상황에 마주할 것으로 예상되는 경우 준비를 해도 늦지 않다.

코딩교육

강의실, 학습 등의 다른 후보단어들이 있었다. 그러나 교육으로 선택한 것은 교수자와 학습자 간의 소통의 통로를 열고 싶기 때문이다. 소통의 장에서는 학습한 내용에 대해 토론하고 학습하는데 교수자가 아니더라도 서로 물어볼 수 있고

플랫폼

서비스라는 말이 있지만 플랫폼으로 규정한 것은 언제든지 변신을 해야하기 때문이다. 최근의 프로그래밍의 추세만 보아도 객체지향형 프로그래밍에 함수형 프로그래밍이 속속들이 등장하고 있다. 지금까지는 60, 70년대에 만들어진 기계를 더 빠르게 운용하고 코드를 더 쉽게 쓸 수 있는 방향으로 발전해왔다. 함수형 프로그래밍이 한켠에 발전을 거듭하다 이제 주류 언어계에 포함되기 시작했다. 이 변화는 프로그래밍을 실제로 하기 전 사고 단계의 변화이기 때문에 코딩교육의 커리큘럼의 많은 부분을 고쳐야 할 수도 있다.
매년 한가지 프로그래밍 언어가 나온다고 하지만 최근의 주류로 등장한 언어들은 사업자의 독점적 지위 등에 의해 알아두어야 할 만한 가치가 있다. 우리가 주로 사용할 Python 으로도 보통 쓰는 플랫폼의 어플리케이션(웹, 안드로이드, iOS, 윈도우)을 만들 수 있지만, 결국 사용자는 자신의 프로그램을 만들기 위해서 또 다른 언어를 찾아야하는 등 유연한 대처를 해야 한다. 우리는 이 변신에 앞으로 동참하기 위해서 플랫폼이라는 단어를 사용하기로 했다. 고정된 하나의 공간보다 언제든지 개선을 할 수 있고 그 개선이 다른 부분에 큰 영향을 주지 않는 공간으로 만들어야 한다. 마이크로서비스 아키텍쳐, 애자일 프로세스의 철학 등 다양한

기술변화에 대응하기 위한 방법을 차용하고자 한다. 또한 이 과정들을 기록하고 문서화 시키

다른 사람들이 후에 이 플랫폼의 생존에 기여할 수 있게 만들고자 한다.

Khan + Udacity

공공 코딩교육 플랫폼의 모습

코딩교육은 학생들이 컴퓨터로 교과 과제를 수행 할 수 있게 한다.

코딩을 모르는 친구들은?

코딩 교육이 없는 사람들을 위해 온라인으로 강의를 듣고 실습할 수 있게 해준다. Udacity와 유사하게 아주 협소한 주제로 간단한 실습을 웹브라우져 내에서 할 수 있게 해준다. 학교 등과 협의하여 방과 후 휴식 시간 등에 컴퓨터를 사용할 수 있게 해준다.

코딩에 필요한 것들은 온라인으로 모두 제공하자

코딩에 필요한 것은 우선 프로그램 언어가 실행 될 수 있는 컴퓨터. 프로그램 언어를 편집할 수 있는 편집기, 코드가 저장되고 변경내역이 관리 될 수 있는 저장소이다.[10]플랫폼(웹사이트)에 로그인 하면 자신이 만들었던 코드들을 볼 수 있고 과거 변경 내역들을 살펴 볼 수 있게 한다.

- 편집기: 보통 현장에서는 IDE라는 이름으로 편집, 실행, 디버깅(에러 잡기)까지 할 수 있는 도구들을 말한다. 그러나 가격이 지나치게 비싸다는 단점이 있다. Python의 경우 PyCharm이 있는데 community edition을 사용하면 무료로 사용할 수 있다. 그러나 처음 사용하는데 있어서 복잡하다

- 변경 내역 관리: 코드가 저장이 안되거나, 변경한 것이 틀렸을 때 되돌릴 수 있는 도구로, 형상관리도구라고도 한다. Git이라는 웹서비스를 통해 누구나 무료로 이 기능을 이용 할 수 있으나, IT 분야에 종사했던 사람이어도 처음 사용하는 사람이라면 꽤 어려운 편이다. 우리나라의 큰 업체도 Git이 좋은 것은 알지만 솔루션을 변경하는데 있어 위험요소가 더 크기 때문에 바꾸는 것을 꺼려한다.

웹브라우저를 고집하는 이유

누구나 사용할 수 있는 도구들이 있지만 코딩교육 플랫폼에서 익숙해 질 수 있게 한다. 이는 본인의 컴퓨터가 아닌 환경에서도 쉽게 접근 할 수 있게 해준다. 물론 이 유용한 도구들을 사용 할 수 있도록 따라하기(Tutorial)등을 폭넓게 제공할 예정이다.

교과과정 연계 프로그램: 세금실험

경제 시뮬레이션을 통한 세금 실험하기 - Random Data를 활용하여

두 집합에 같은 수의 사람이 있으며, 소득은 정규분포에 의해 Random 으로 부여한다.

매년 1회 세금을 징수하는데 두집합의 세율은 차이가 난다.

(Optional) 세율을 누진세로 한다.

(Optional) 그래프 또는 엑셀 등 다른 프로그램에서 쓸 수 있게 데이터를 내보내기(Export)

5년 10년 등 징세 횟수가 지남에 따라 두집합의 소득 격차가 어떻게 차이나는지 지니 계수를 산출한다

필요 지식

- 코딩플랫폼 측

다수의 개인 데이터를 저장 해 둘수 있는 Dictionary 형 데이터 다루기 난이도 하

(Optional) 100만 세트 이상의 데이터를 빠르게 다루기 위한 라이브러리(외부기능)사용법. (난이도 중)

매년 개인의 소득이 늘어나고, 세율에 의해 징수되는 순서도 또는 의사(Pseudo) 코드 등 작성 (난이도 중)

순서도나 의사코드를 실제로 구현하기 위한 반복문 과 흐름제어 (난이도 하)

(Optional) 차트로 그리거나 엑셀 등에서 사용할 수 있게 Export 함(난이도 중)

(Optional) Markdown 코드를 활용하여 보고서를 코드로 직접 만들고 PDF 문서 등으로 배

포함 (난이도 중)

- 교과 교육 측

분위별 소득 격차와 지니계수 (사회, 경제)

정규분포에 대한 개념 이해 (수학)

세율을 함수로 표현하기 (수학)

지니계수 계산(사회, 경제)

데이터로 인포그래픽스 만들어보기 (미술, 수학, 국어)

수업 설계

- 코딩플랫폼 측

이 프로그램을 진행하고자 하는 학교의 학생들의 코드 이해도를 알아보기 위해 사전지식 테스트. 성적에 반영되는 것을 막기 위해 개별 데이터는 제공하지 않음

수업 진행에 따라 어떤 예제를 주어줘야 하는지, 어떤 개발 환경을 제공해야 하는지 준비

예제를 수행하는데에 있어 자주 범할 수 있는 실수 등을 검출 할 수 있는 테스트 시나리오 준비

- 교과 교육 측

사회, 경제 교과 교사 2인 이상이 모여 현재 학생들의 학업 성취도에 대해서 의논한다. 위에

서 제시한 프로그램에 해당하는 지식이 어떤 것이 필요한지 판단함. 학생들의 수준, 수업 시수 등의 여건에 에 따라 개인 또는 팀 등 수행 단위를 설정함

처음부터 기초 코딩 교육을 실시해 학생들이 자신만의 경제 시뮬레이션 코드를 만들 것인지 또는 이미 만들어져 있는 코드를 활용 할 것인지 결정함. 또한 단순히 적절한 숫자를 보여주는 것 부터 차트, 그래프, 보고서, 배포 까지의 단계에서 어느 정도 수준까지 진행 할 것인지 결정함

프로그램을 진행하는 데에 있어 어느 부분을 평가에 적용 할 것인지 정의함

코딩 또는 교과 지식이 취약한 학생들을 이끌고 갈 방법, 컴퓨터에 대한 접근성이 떨어지는 취약계층 학생 지원 방안

코딩을 하기 전에

콘솔과 친해진는 방법 부터 배우기

과거 도스 와 같이 직접 명령어를 입력해서 컴퓨터를 다루는 방법. G-UI과 구분하여 CLI(Command Line Interface)라고 함. 대부분의 운영체제는 GUI로 작동하지만 검색어 URL 아이디와 비밀번호 와 같이 네트워크에서는 대부분 아직도 CLI 방식을 이용하고 있다. 코딩은 GUI 뒤에 숨겨져 있는 명령어들을 찾아가는 과정으로 영문으로 가득한 답답한 화면을 마주한다. 그러나 실제 컴퓨터를 만나기 위해서는 이것 부터 시작해야 한다.

관행에서 벗어나기

우리나라의 컴퓨터 환경은 WINDOWS, MS OFFICE, 한글20XX, 인터넷 익스플로러, 네이버로 한정되어 있는 경우가 많다. 특히 자료 검색에 있어서 코드관련 된 자료의 검색은 네이버와 구글의 검색결과가 심하게 차이난다. 그렇기 때문에 과제를 이행하는데 있어 폐쇄적 검색 결과를 보여주는 네이버에 의존해서는 안된다. 또한 인터넷 세계의 다양한 요소를 파헤치지 위해서는 인터넷 익스플로러 보다는 Firefox, Chrome 등을 써볼 필요가 있으며, 대부분의 서버는 Linux 환경에서 구현되기 때문에 WINDOWS 마저 벗어날 필요가 있다.

세계의 개발자들과 소통할 준비

개발 언어가 영어로 되었다고 하지만 이는 자연어로서 영어를 본떳다기 보다. 수학을 표현하는 영어를 옮겼다고 볼 수 있다. 즉, 영어를 사용하지 않아도 충분히 코드를 이해할 수 있으

며 약간의 영어를 통해 질문을 구하고 답변을 해줄 수 있다. (간혹 코드로만 답변을 하는 경우가 있다) 우리말로도 여러 검색 결과가 나오기는 하지만 약간의 수준이 올라가면 대부분 영어로 소통한다. 이는 실전 영어를 사용해볼 기회이기도 하다. 개발자는 코드가 이상하다고 말할 수 있으나, 질문의 영어가 문법이 잘못되었다고 하지 않는다. 전 세계 개발자가 대상이기 때문에 쉬운 영어를 쓰는 것이 매너이기도 하다.

이것은 지킵시다

주입식 교육 금지

늘 우리나라는 주입식 교육이 문제다. 언어들이 아무리 쉽고 좋아져도 코드를 작성하는데에 몇가지 관행이 있어서 코드를 읽기 편하게 해준다. PyCON 2017 의 <초중고 코딩교육> 토론 세션에서 한 특성화고 학생은 "시험은 어떻게 봐요?"라는 질문에 이렇게 대답했다.

> — 일단, 시험은 시험지에 보는데(일동 웅성거림) 그 명령어들이 알듯 말듯하고. 어짜피 선생님들도 찾아서 보여주는데 우리는 외워야 되고. 아차, 시험문제에 카멜케이스(코딩 관행)을 지키지 않으면 점수가 까여요.

명령어를 외우고 관행을 지키는 것은 프로그램을 짜는데에 아무 도움이 되지 않는다. 명령어가 헷갈리면 찾아주면 된다. 그리고 대부분의 프로그래밍 도구(IDE 등)들은 몇개 단어만 치면 명령어 후보들이 나와 자동으로 완성이 된고 실행 전에 문법을 검사해준다.

주입식 교육에는 분명한 목적이 있다. 목적은 대체로 시험이다. 시험 이후의 목적은 현실과 거리가 멀다. 주입식 교육으로 지식을 오래 기억하고 있어도 현실에 자주 부딪히지 않으면 그 속에 담긴 의미를 알지 못한다. 특히, 프로그래밍 분야는 명령어들을 외우거나 형식을 지켜야 하는 제약이 많아 주입식으로 흘러갈 가능성이 크다. 그러나 우리가 코딩교육을 하는 제 1 의 목적으로 돌아가보자. 한 사람이 살면서 마주하는 문제를 프로그래밍 또는 프로그래밍 하듯 풀어나가게 도와주는 것이다. 형식, 기법 등을 주입하는 것 보다 왜 써야하는지, 왜 지켜야하는지 개선할 부분은 무엇인지 가능성들을 생각할 시간을 충분히 주어야 한다.

할 수 있는 데 까지만

필자에게 2010년에 있던 일이다. "우리나라에도 르 꼬르동 뷔지에 코스 있잖아?" 당시 월급도 꽤나 모아뒀고 이왕 하는 취미라면 제대로 해보고 싶어서 어느 요리학원 다니기 보다 세계 일류 과정이라는 곳을 체험해 보고 싶었다. 똑같이 이렇게 가족이나 동료들에게 물어본다면 당시 내가 들었던 대답과 비슷한 대답을 들을 것이다.

코딩을 공부해서 삶이 나아지게 문제를 해결하게 만들어 주는 것 그리고 적성에 맞다면 그 길로 나아가게 도와주는 것이 이 플랫폼의 사용자측 목표상태[11]이다. 그러나 어떤 사람은 본인에게 필요한 수준까지 쓰고 싶어하는데 억지로 끌고 가려는 노력을 기울이면 안된다. 세상을 살아가는 방법에 코딩은 꼭 필요한 것은 아니다.

너무 어려운 문제 금지

미적분을 고등학교 교과과정에서 빼자는 논란이 있었다. 이미 선택과목이 되면서 부터 대학교수들은 비상을 울렸다. 미적분도 모르고 이공대로 입학하면 어떻게 하라는 말이냐는 볼멘소리가 터져 나왔다. Huzen Dev. House는 이 토론에 만약 의견을 제시하라면 미적분은 고등교과에 꼭 필요하다는 입장이다. 미적분을 통해서 세상에 알 수 있는 것들은 너무나 많다. 그리고 공학, 물리학, 수학에만 쓰이는 것이 아니다. 언젠가는 한번 쯤 마주칠 법한 주제다. 문제는 미적분이 어렵다기 보다 시험 문제가 어려운 것이 아닌가? 우리나라의 물리학자도 이와 같은 의견을 낸 적이 있다.[12]

이 교육의 목표는 알고 행동하는 것 그 자체에 있어야한다. 코딩교육에도 아는지 모르는지 시험도 볼 필요가 있고 다소 어려운 개념을 가르칠수도 있다. 하지만 이 모든 것은 삶을 나아지게 만드는 것이지 성적과 오지도 않은 미래에 대한 준비는 아니다.

사업모델

일단 오픈 부터

가장 중요한 부분은 역시나 어떻게 유지하는가에 달려있다. 서비스 플랫폼은 유지할 수 있는 재정적 뒷받침만 된다면, 간단하게 최소한의 규모로 시작해서 사용자의 요구사항을 들어가며 덩치를 키울 수 있다. 최초 서비스에는 Python 의 기초 수업, 실습, 그리고 2개의 교과 과정 연계 프로그램을 제공한다.

재정 수급

재정 수급의 원칙: Somebody pays (참조: Jesse Noller, Pycon: EveryBody Pays [13]) 모두가 지불할 수 없는 경우가 많으므로 누군가는(Somebody) 대신 부담하는 원칙으로 변형했다.

1) 비영리 재단으로 운영: 정부, 학교, 교사, 학부모, 단체 등으로 부터 후원을 받는 방법
2) 상업용 서비스: 오픈소스를 포기하고 처음부터 학교, 교육청 단위로 가격을 제시하고 사용 인원에 따라 가격을 책정한다.

조직 운영

- 인력 운용의 규칙: 모두가 자신의 일만큼 수익을 거두어야 한다.
- 플랫폼 제작인원 4명(기획, Project Manager 포함), 조직 운영 및 대외 홍보 매니져 2명, 팀장급 2명(제작, 운영)

- 장기 계획으로 운영인력 모집. 프리랜서 중 수익을 공유하며 초반 플랫폼 제작 작업에 동참 할 수 있는 인력부터 운용함
- 학교 교육에 바로 쓰일 수 있는 형태로 제작하여 누구나 한번쯤은 거치게 하여 인지도를 높임
- 학교를 졸업하고 전일제로 참여하고 싶은 엔지니어를 장학생으로 활용 (열정페이 방지)

열린 플랫폼으로

Github 등 의 서비스를 통해 소스를 공개하고 개선에 참여할 수 있는 창구를 만든다. 단순히 개선사항을 게시판, 이메일 등으로 제보하는 것이 아니라, 공개된 소스를 보고 직접 개선 코드를 제출 할 수 있게 한다. 일종의 Wiki 서비스 방식이다. 현재의 코드를 보고 무엇이 문제인지 또는 개선 방식이 적합한지 토론해야한다. 토론 후 적합여부에 따라 해당 수정사항을 반영한다. 적합 여부는 수정 코드가 다른 서비스에 영향을 주지 않는지[14] 충분히 테스트를 거친 후 결정한다. 이를 위해 TDD방식으로 프로젝트를 진행하고자 한다.[15]

사실은

생활코딩

우리나라에도 이에 못지 않은 훌륭한 플랫폼이 있다. 이고잉(닉네임)의 www.opentutorials.org 가 일반인을 대상으로 다양한 강좌를 영상으로 시연하는 모습을 보여준다. 이외에 다양한 컴퓨터 프로그래밍 이론들을 올려놓기도 했다. 생활 코딩은 오픈튜토리얼스라는 비영리 재단으로 탈바꿈하여 회원들의 후원금으로 운영되고 있다.

강의 품질이 상당하여 현장 개발자들도 어떻게 이 많은 분야를 자세하게 아느냐는 댓글도 달린다. 그래서 개발자들이 간혹 '생코신공'이라는 말을 하기도 하는데 생활코딩에서 검색해서 (일을) 풀어냈는 뜻이다. 아마도 초기의 모양새는 생활코딩의 서비스와 비슷하게 만들어 질 것 같다. 오히려 이 서비스로 공부를 하고 스터디 모임을 만들어 주는 대안도 있다. 그러나 해당 사이트는 수업의 폭이 넓어 어디서 부터 시작해야하는지, 시작하는데 (개발환경구축, 테스트 서버 구축 등) 어려움을 겪는 등 초기에 따라잡는 시간이 많이 걸리고, 질문-답변에 한계가 있다.

다시 Udacity 스타일이 필요한 이유

Udacity는 제공하는 서버에서 코드를 입력하고 검증을 받는 식으로 진행이 되어서 자신이 짠 코드가 맞는지 확인하는데 시간이 얼마 안걸린다. 그리고 정확한 목표를 그때 그때 지정해 주기 때문에 좋은 가이드라인을 제공한다. 그래더 더욱 공공 코딩교육 플랫폼은 피드백이

빠른 Udacity의 방식이 필요하다.

4. 두려워 할 필요가 없는 세상

기술변화를 감지하는 눈을 키워주자

알파고가 우리 사회를 지배할까?

알파고가 이세돌을 4:1로 이기고 난뒤 반응은 충격적이었다. 그런데 기술계 일부는 알파고보다 이세돌을 더 무서워했다. 최첨단 기술로 무장한 계산 기계를 인간이 이겨버린 것이다. 이를 보면 미래는 암울해 보이지 않는다. 반대의 사례가 있다. 두개의 인공지능 의사소통 기기를 두고 서로 대화를 하게 하니 인간이 알아들을 수 없는 언어로 소통을 하기 시작했다. 이를 보면 다시 미래는 암울해 보인다. 이 두가지 사례에서 공통적으로 놓친 부분들은 이 기계들은 프로그램상 한가지의 목적을 가지고 있다는 점이다. 기술적 시각을 틔운 사람들에게는 이 현상이 그렇게 암울하지도 밝아보이지도 않으리라 생각한다. Huzen Dev. House 는 이 현상들이 어떻게 구현했고 어떤 장비들이 사용되었을지 호기심을 자극하는 소재로 보인다. 그러나 신문 보도, 교양 도서에서만 보면 우리의 미래를 이미 예측한 듯 글을 써내려간다. 십수년 전에는 경제학, 사회학분야가 그랬었다. 곧 세상의 지하자원이 고갈될 것이다. 경제체제가 붕괴하고 전쟁이나 혁명이 일어날 것이다. 반대로는 한국이 세계의 경제의 중심이 될 것이다. 주식 시장은 2000포인트를 훌쩍 넘길 것이다. 이런 학자들의 발언이 사람들의 심리를 자극했다. 이미 세계대전 이후 부터 1984같은 소설이 쓰여졌고, 영화만 봐도 매드맥스, 터미네이터 같은 디스토피아물이 그 심리를 대변했다. 하지만 인간은 극단적인 상황에도 현명한 방법을 찾아냈었고, 극단적인 선택을 섣불리 하지도 않았다. 아직 지하자원은 고갈되지

않았고 사회적으로 해결할 방법들을 모색하고 있다. 물론 학자들의 발언은 "이대로 가다 간…"에 대한 경고이다. 그러나 보통의 사회에 전달되는데 많은 왜곡을 거쳐 전달이 되었다. 이런 현상은 인터넷으로 정보가 기존의 매체와 기성 매체에 종사하는 사람들을 통하지 않고 전달되면서 많은 부분 해소되었다. 인터넷이라는 매체 이전에 사람들이 사회, 경제 변화의 맥락을 읽는 눈이 생겼기 때문이다.

무인자동차와 대리운전

세상의 일은 예측하기 어렵다. 50년 전에 이미 자동 운전 기계(무인자동차)의 발상이 나왔지만 아직도 그 문제는 해결되지 않았다. 하지만, 대리운전이라는 직업이 생겼다. 기술의 시각에서 보면 오히려 시대를 역행하는 발상이지만 우리의 생활에 매우 유용하게 만들어주었다. 세탁기, 냉장고 IoT 가전제품들이 속속들이 나와있지만 심부름센터는 사라지지 않고 더 호황을 누리고 있다. 이런 세상에 알파고가 과연 세상을 지배하는 날이 도래할까?

아무도 모른다. 하지만 우리는 당장 어느날 알파고가 등장해서 그 지배를 받지는 않는다. 하나씩 변화가 일어나는 것을 감지하고 이에 대해 시민 사회의 대화(담론)를 형성해 나가면 우리가 생각하는 비극은 그렇게 암울하지는 않을 것 같다. 코딩교육을 통해 우리는 이 시대의 변화에 그 내면을 들여다 보는 시각을 키우고자 한다. 예를 들어본다면, 트랜지스터를 여러 개 연결해서 2 ~ 4비트 수준의 간단한 컴퓨터를 만들 수 있다. 이를 통해 CPU 라고 하는 장치는 이 전선들이 나노크기로 배열되어있고 여기에 전압이 걸리냐 걸리지 않느냐로 작동하는 것을 알게 된다. 행렬 연산을 통해 일부 데이터에 반응하는 간단한 인공지능을 만들 수 있

다. 알파고와 같은 기계들은 이 간단한 장치의 원리를 다수 계층으로 더 크고 빠르게 만들었을 뿐이다. 세상의 변화는 인간이 따라갈 수 없는 수준이 아니라 인간의 무관심에 의한 것으로 생각한다. 코딩교육은 그 기술변화의 맥락을 읽는 눈을 키워줄 것이다.

Ends

---

[1] 보통 프로그래밍 언어의 분류는 전자신호로만 구성된 기계어, 다음으로 기계어를 처음으로 인간의 언어로 번역한 어셈블리, 그리고 어셈블리언어를 구조적으로 쓰는 Native(C, C++, Fortran, JAVA), Native 를 효과적으로 쓰기 위한 고급언어(Python, C#, Rust, Go 등)로 분류한다.

[2] PyCon 2017 <초중고 코딩교육>세션, 2017년 8월13일

[3] 코딩을 할 때 사용하는 에디터마다 색이 다르게 표현된다.

[4] 퍼블리싱 또는 배포한다고 한다.

[5] 구글은 사내에서 개발자들이 무분별하게 여러 언어를 사용하는 것을 방지하기 위해 공식 언어를 두고 있다. 최근 Go 라는 구글에서 개발한 언어가 추가되었다. 애플도 비슷하게 swift 라는 자사가 개발한 언어를 이용해야 iOS용 어플리케이션을 제작할 수 있다.

[6] 실무에서 디버깅이라는 작업을 하게 되는데 가장 먼저 마주하는 에러는 문법상 오류인 오타, 괄호 불일치, 쉼표(콤마), 콜론과 세미콜론 탈락 등이 가장 먼저 발생한다. 물론 IDE에서

자동 완성 기능을 통해 사전에 방지 할 수 있지만 한계가 있다.

[7] 과거의 바이러스 코드는 빠르게 감염을 시켜야 하기 때문에 30라인도 안되는 코드로 이루어진 경우가 대부분이다. 물론 최근의 일부 바이러스들도 몇안되는 명령어로 컴퓨터를 마비시킨다. 다행히도 백신도 그에 못지 않게 간단하게 만들 수 있다

[8] 수학교과서 맨뒤의 로그표 등을 참고하는 법도 있지만 요즘은 계산기를 활용하는 것이 실제 생활에도 더 도움이 된다.

[9] 직역을 하면 정보과학자인데 통계학자, 인공지능학자, 알고리즘학자 등이 이 직업에 포함되기도 한다.

[10] 보통 코딩 하기 전 편집기(IDE)와 인터프리터(프로그래밍 언어에 따라 컴파일러) 형상관리 도구를 설치하는 작업을 작업(개발)환경구축이라고 한다.

[11] User Target-Status 서비스를 이용해야할 이유를 제공하는 것, 서비스를 이용한 후 변화하게 되는 상태, 사용 효과, 효익Benefit 등으로 해석가능

[12] 이종필 교수(건국대) <아인슈타인탄생 100주년 기념 이종필 박사의 아주 특별한 상대성이론 강의>

[13] http://jessenoller.com/blog/2011/05/25/pycon-everybody-pays

[14] 리팩터링Refactoring

[15] TDD 방식: 프로젝트 시작을 제작 보다 테스트 중심으로 구축하는 방법.