Final Project

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

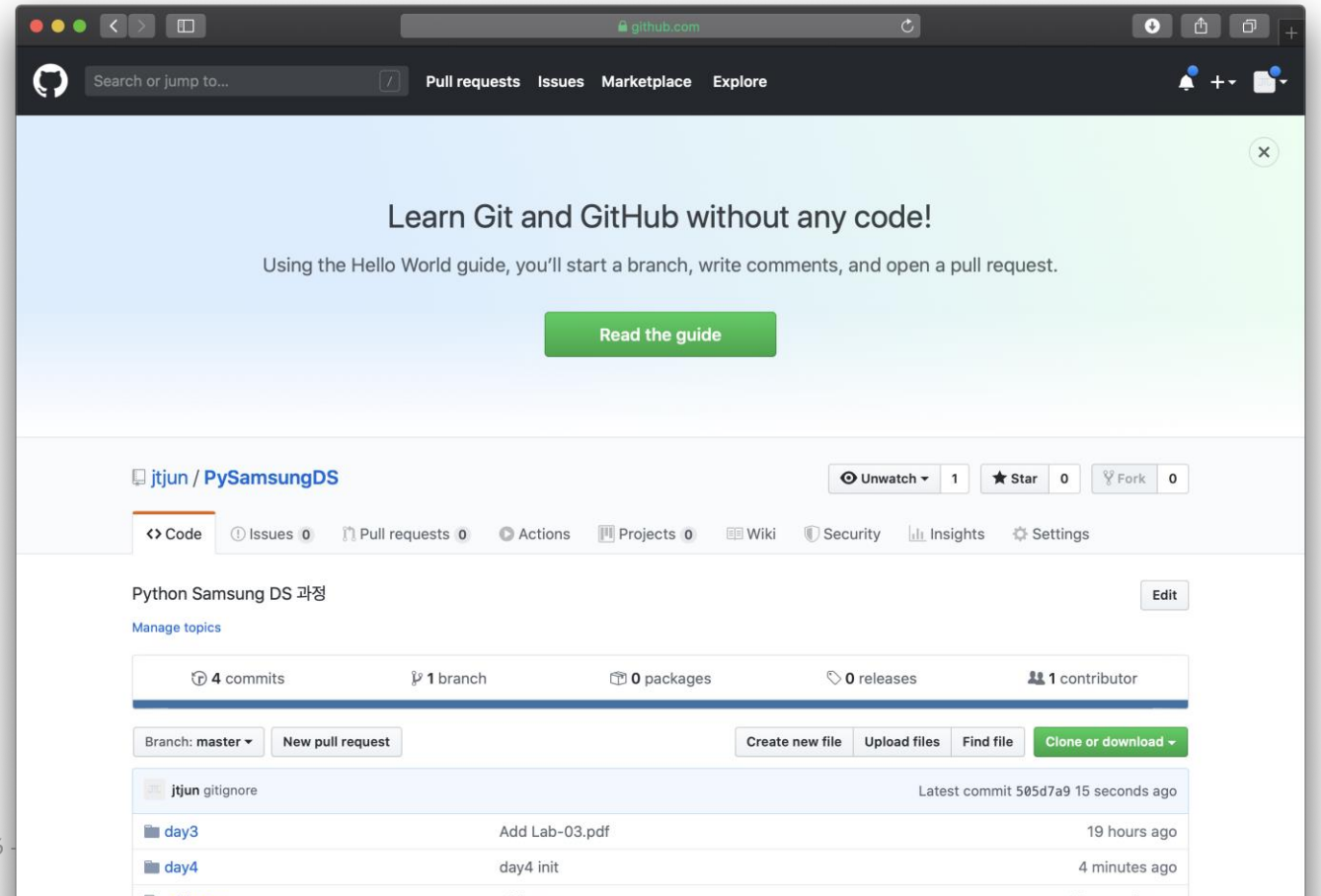
Systems Software &
Architecture Lab.

Seoul National University

Dec 16 – 20, 2019

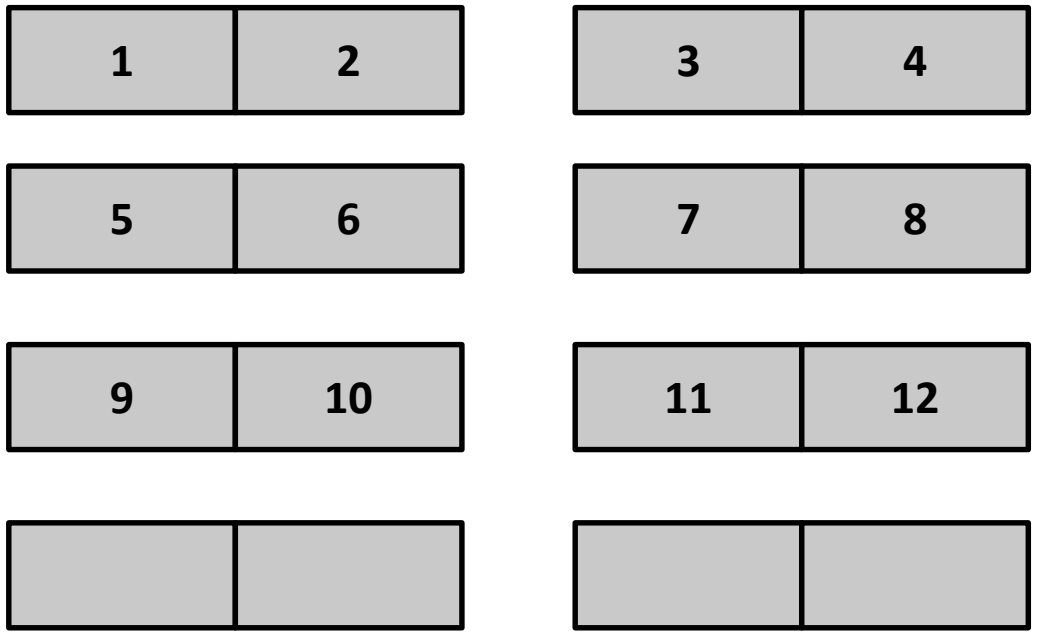
Project. 다운로드 안내

- https://github.com/jtjun/PySamsungDS
- https://github.com/jtjun/PySamsungDS/archive/master.zip



Project. 팀 구성

- 랜덤하게 구성하였습니다.
- 팀원끼리 함께 앉아주세요.



Schedule

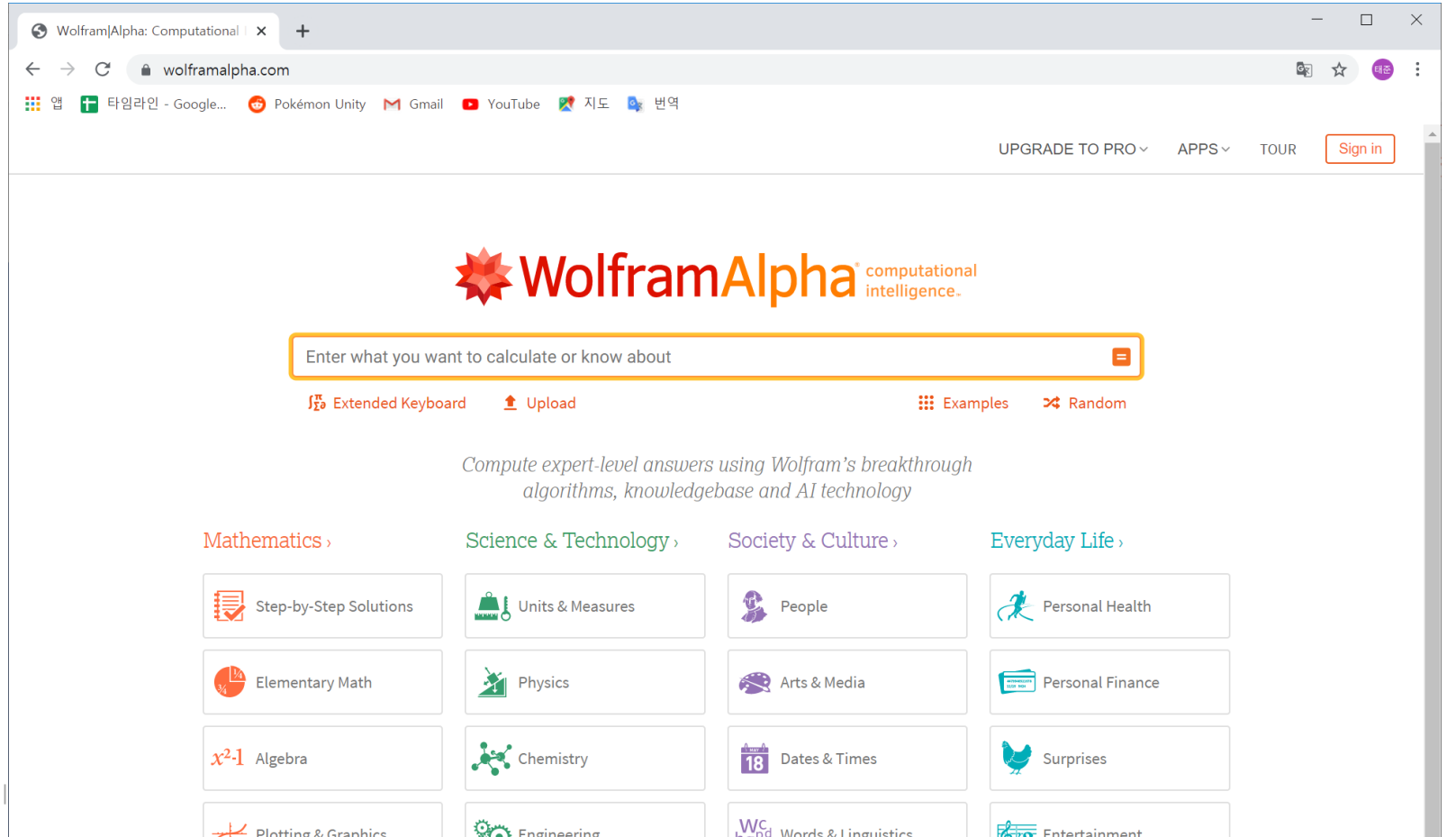
Project. 일정

- 08:00 조 편성 공지 & 뼈대 코드 다운로드
- ~09:00 Project. Wolfram-Beta 스펙 설명
- 12:00 점심시간
- ~12:00 팀별 input 파일(공격) & output 파일(방어) 받기
- 13:00 받은 공격 파일 내용 발표
- ~16:00 코딩 시간
- ~16:40 발표
- ~17:00 Project 마무리
- 17:00 귀가 예정

Wolfram-Alpha

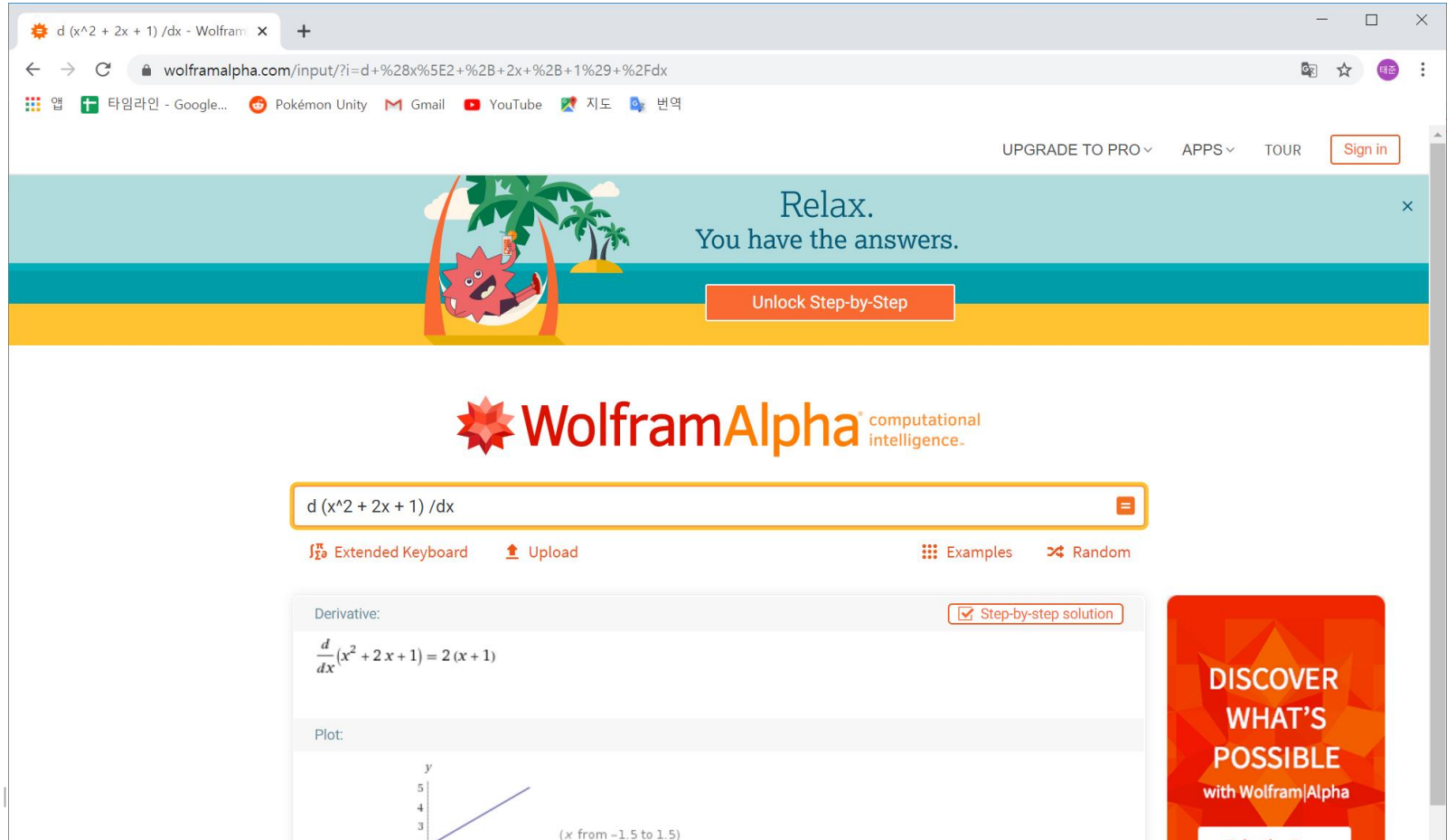
Project. Wolfram-Beta

- Wolfram-Alpha
  - https://www.wolframalpha.com/



Project. Wolfram-Beta

- Wolfram-Alpha
  - https://www.wolframalpha.com/



Wolfram-Beta

Project. Wolfram-Beta

- 최종적으로 만들고자 하는 그림
  - We can do it!!!

```
$ cat output.txt↵
D,x^2 + 2x + 1
I,2x + 2,1
C,x^3 + x^2 + x + 1,-1
$ python wolfram_beta.py↵
$ cat output.txt↵
2x + 2
x^2 + 2x + 1
0
```

Project. Wolfram-Beta

- 최종적으로 만들고자 하는 그림
  - 입력 파일에 있는 요청을 모두 처리하여 출력 파일 만들기
  - 입력 요청의 종류 : 3가지

input.txt

```
D,f(x)
I,f(x),c
C,f(x),a
```

  - 미분(Differential), 적분(Integral), 계산(Computation)

output.txt

```
f'(x)
F(x) + c
f(a)
```

Project. Wolfram-Beta

- 구체적인 제약조건
  - **다항함수 문자열**
    - 다항함수 문자열은 최소 1개 이상의 항으로 구성되어 있어야 한다.
    - 각각의 항은 덧셈(`'+'`) 기호로 구분되어 있으며, 덧셈 기호 사이에는 한 칸의 공백이 있다.
      - `'[first_term] + [second_term] + … + [last_term]'`
    - 항은 (정수) **계수**와 (음이 아닌 정수) **차수**로 표현할 수 있다.
      - 차수가 0인 경우, 이 항을 **상수항**이라 하며, 정수 계수를 있는 그대로 출력한다. (예: `'-1'`, `'0'`, `'1'`)
      - 차수가 1 이상이면, 정수 계수 뒤에 문자 x를 붙인다.
        추가로, 차수가 2 이상이면, 문자 x 뒤에 기호 ^와 차수를 붙인다. (예: `'-5x^2'`, `'0x'`, `'3x^4'`)
      - 차수가 1이상이고 계수의 절댓값이 1인 경우, 숫자 1을 생략한다. (예: `'-x'`, `'x^2'`, `'-x^3'`)

Project. Wolfram-Beta

- ‘D,f(x)’ 구체적인 제약조건
  - 입력 받은 **다항함수 문자열**을 (입력 파일로 부터)
    ‘미분한 결과’를 **다항함수 문자열**로 출력하기 (출력 파일로)

  - **파일을 line 별로 해결**
    - Input 파일에서 ‘D’ 로 시작하는 줄은 미분을 요청함
    - 같은 식으로 ‘I’로 시작하는 줄은 적분을 요청,
      ‘C’로 시작하는 줄은 계산을 요청
    - Output 파일의 같은 번째 줄에 해당 요청의 결과를 기록

Project. Wolfram-Beta

- '`D,f(x)`' 구체적인 제약조건
  - 미분함수
    - 어느 임의의 다항 함수의 미분 함수는 다항 함수이다.
    - → **미분 함수 역시 여러 다항함수 문자열**로 표현 가능하다.

  - 예 : equation = `'x^2 + 2x + 1'`
    - `'2x + 2 + 0'`
    - `'2x + 2'`
    - `'2 + 2x'`
    - `'1 + x + 1 + x + 0x^2 + 0x^3'`

  - 채점 코드는 항의 순서, 계수가 0인 항, 동류항 묶기 등등을 신경 쓰지 않는다!
  - 신경 쓸만한 부분은, 다항함수 문자열은 **최소 1개의 항**이 필요하다는 점
    - 빈 문자열 `''`을 출력하면 안 된다. 최소한 `'0'`은 출력해야 함

Project. Wolfram-Beta

- `def print_term(degree, factor):`
  - *degree* : 항의 차수 (0 이상의 정수)
    - 0인 경우?
    - 1인 경우?
  - *factor* : 항의 계수 (정수)
    - 음수인 경우?
    - 절댓값이 1인 경우?

```
assert print_term(2, 0) == "0x^2"
assert print_term(1, -1) == "-x"
assert print_term(0, 5) == "5"
```

Project. Wolfram-Beta

- `def print_equation(terms):`
  - *terms* : dict
    - dictionary {degree (int) : factor (int)}의 형태
    - key := degree (차수), value := factor (계수)
  - Expected output
    - `str` 함수를 문자열로 표현 ←`' + '` 문자를 기준으로 합침 : `string.join()`?

```
assert print_equation(
  {2:0, 1:-1, 0:5}
) == "0x^2 + -x + 5"
```

Project. Wolfram-Beta

- `def parse_term(term_str):`
  - `print_term()`의 역함수 꼴
  - *term_str* : 항을 문자열로 표현

  - Expected outputs
    - tuple : = (degree : int, factor : int)
    - (차수, 계수) 순서쌍

```
assert parse_term("0x^2") == (2, 0)
assert parse_term("-x") == (1, -1)
assert parse_term("5") == (0, 5)
```

Project. Wolfram-Beta

- Dictionary 의 key 와 value
  - Dictionary 의 key 는 str 뿐만 아니라 int, float, tuple 도 가능. 단, list는 불가능
  - Dictionary 의 value 는 list도 가능

```
>>> d = dict()↵
>>> d['my_key'] = 'my_value'↵
>>> d[3] = 3.21↵
>>> d[10.16] = [21, 'to', 22]↵
>>> d[('lat',1.25,(6,-2.0))] = 'tuple'↵
>>> d[[1,2,3]] = 'error'↵
TypeError: unhashable type: 'list'
```

Project. Wolfram-Beta

- `def parse_equation(equation):`
  - *equation* : str
    - `' + '` 문자를 기준으로 쪼개기 : `string.split()`?
  - Expected output
    - dictionary {degree (int) : factor (int)}의 형태
    - key := degree (차수), value := factor (계수)

```
assert parse_equation("0x^2 + -x + 5") == \
    {2:0, 1:-1, 0:5}
```

Project. Wolfram-Beta

- `def d_dx_as_terms(terms):`
  - *terms* : dict
    - dictionary {degree (int) : factor (int)}의 형태
    - {1항 차수 : 1항 계수, 2항 차수 : 2항 계수, …, N항 차수 : N항 계수}
  - Expected output
    - 형식은 *terms*와 동일
    - 가능한 답은 여러가지가 될 수 있음!
    - 신경 쓸 부분 (**다항함수 조건**)
      - 출력된 계수는 정수인가
      - 출력된 차수는 0 이상의 정수인가
      - 항의 개수는 1 이상인가
      - 미분 법칙에 잘 맞는가 : $(\frac{d}{dx}ax^n = (an)x^{n-1}, \frac{d}{dx}a = 0$ and $\frac{d}{dx}(f+g) = \frac{d}{dx}f + \frac{d}{dx}g)$

Project. Wolfram-Beta

- `def d_dx(equation):`
  - *equation* : str
  - Expected output : str

  - 지금껏 구현했던 함수들을 총동원하기
  - 3줄로 간결하게 표현할 수 있음

```
assert d_dx("0x^2 + -x + 5") == "2x + -1"
```

Project. Wolfram-Beta

- ‘I,f(x),c’ 구체적인 제약조건
  - 입력 받은 **다항함수 문자열**을 (입력 파일로 부터)
    ‘적분한 결과’를 **다항함수 문자열**로 출력하기 (출력 파일로)

  - ‘c’
    - 적분 상수
    - 정수 (int)

Project. Wolfram-Beta

- ‘I,f(x),c’ 구체적인 제약조건
  - 적분함수
    - 어느 임의의 다항 함수의 적분 함수는 다항 함수이다.
    - → **적분 함수 역시 여러 다항함수 문자열**로 표현 가능하다.

  - 예 : equation = ‘2x + 2'
    - ‘x^2 + 2x + c'
    - ‘x^2 + x + x + c'
    - ‘2x + x^2 + c’
    - ‘c + x + x^2 + x + 0x^3'

  - 채점 코드는 항의 순서, 계수가 0인 항, 동류항 묶기 등등을 신경 쓰지 않는다!
  - 신경 쓸만한 부분은, 다항함수 문자열은 **최소 1개의 항**이 필요하다는 점
    - 빈 문자열 ''을 출력하면 안 된다. 최소한 '0'은 출력해야 함

Project. Wolfram-Beta

- `def integral_as_terms(terms, constant):`
  - *terms* : dict
    - dictionary {degree (int) : factor (int)}의 형태
    - {1항 차수 : 1항 계수, 2항 차수 : 2항 계수, …, N항 차수 : N항 계수}
  - Expected output
    - 형식은 *terms*와 동일
    - 가능한 답은 여러가지가 될 수 있음!
    - 신경 쓸 부분 (**다항함수 조건**)
      - 출력된 계수는 정수인가
      - 출력된 차수는 0 이상의 정수인가
      - 항의 개수는 1 이상인가
      - 적분 법칙에 잘 맞는가 : $(ax^n = \int (an)x^{n-1}dx,\ a = \int 0,\ \text{and}\ \int (f+g) = \int f + \int g)$

Project. Wolfram-Beta

- `def integral(equation, constant):`
  - *equation* : str
  - *constant* : int
  - Expected output : str

  - 지금껏 구현했던 함수들을 총동원하기
  - 3줄로 간결하게 표현할 수 있음

```
assert integral("2x + -1", 5) == "x^2 + -x + 5"
```

Project. Wolfram-Beta

- ‘C,f(x),a’ 구체적인 제약조건
  - 입력 받은 **다항함수 문자열**을 (입력 파일로 부터)
    함수로 전환해 ‘**정수 a 를 대입한 결과**’를 **정수**로 출력하기 (출력 파일로)

  - ‘a’
    - 0이 아닌 정수

Project. Wolfram-Beta

- `def compute(equation, x):`
  - *equation* : str
  - *x*: int (!=0)
  - Expected output : int

  - 지금껏 구현했던 함수들을 총동원하기
  - 구현한 함수들로 간결하게 표현할 수 있음

```
assert compute("2x + -1", 5) == 9
```

Project. Wolfram-Beta

- `def solve_query(line):`
  - *line* : str
  - Expected output : str

  - 현재, 처리하고 있는 줄 '`line`'이 어떤 요청인지 판단해서
  - 그에 대응하는 응답을 반환함
  - 이미 구현한 함수를 이용하면 간결하게 표현할 수 있음

```
assert solve_query("D,0x^2 + -x + 5") == "2x + -1"
assert solve_query("I,2x + -1,5") == "x^2 + -x + 5"
assert solve_query("C,2x + -1,5") == 9
```

Project. Wolfram-Beta

- 파일 쓰기

```
f = open("./test.txt", "w")
f.write("Hello\n")
f.write("My")
f.write("World!")
f.close()
```

  - 실행 결과

./test.txt

```
Hello
MyWorld!
```

Project. Wolfram-Beta

- `def solve(input_path, output_path):`
  - *path* : str
  - Expected output : None

  - '`input_path`' 에는 1 줄에 1 개의 요청이 적힌 파일이 있음
  - 함수 호출 후 '`output_path`'에 파일 만듦
  - 만들어진 파일의 각 줄에는 입력 받은 파일의 요청에 대한 응답이 있음
  - 즉, 입력 파일 10번째 줄이 '`D,0x^2 + -x + 5`' 라면,
    만들어진 출력파일 10번째 줄은 '`2x + -1`'
  - word_count, day4 실습처럼 파일에서 정보를 읽고 사용하는 모듈

Scoring

Project. Wolfram-Beta

- Test-Cases
  - 총 1000 개의 Test-Case 준비 됨
    설명드린 대로 다항 함수만 존재, 잘못된 입력 없음 (ex 'C,x^-1,0')
  - 뼈대코드 폴더에는 이 중 100개를 샘플링한 Test-Case 존재
    이를 모두 맞추면 100점 확보
  - 각 팀별로 10개의 Test-Case 를 제출 (뒷 장 설명)
    input_team00.txt 파일과 output_team00.txt 파일 한 쌍을 제출
  - 상대 오차, 절대 오차 10^{-6} 까지 허용

Project. Wolfram-Beta

- Test-Cases 공격 & 방어
  - 각 팀별로 10개의 Test-Case 를 제출
    input_team00.txt 파일과 output_team00.txt 파일 한 쌍을 제출

  - Hard Input 을 넣어, +alpha 를 구현한 경우에만 점수를 받을 수 있도록 공격
    - cos(x), sin(x), exp(x)
      - Degree 에 차수(int)가 아닌 함수 이름(str) 이 들어감.
      - type(x) is str -> 문자열인지 아닌지 판별
    - 실수형 계수 (int 아닌 float)
    - 음수 차수 (단 -1 제외)
    - N.5 차수 (sqrt 가능)
    - I 또는 C 요청에서 에서 적분 상수 c, 입력 값 a 에 실수 (float)
    - math 라이브러리 사용 가능 (어제와 동일 import math)

Submission

Project. Submission

- 팀별 테스트 케이스 10개 제출
- input_team00.txt & output_team00.txt 제출
  - jtjun7132@gmail.com

  - 제출 기한 : 12:30
  - 설명 드린 대로 보내주신 input & output 파일을 모아 다른 팀들과 경쟁합니다.

Project. Submission

- `wolfram_beta.py` 소스 코드 제출 *파일 이름 지켜주세요*
- 메일 제목 : 팀 번호
- 메일 내용 : 팀 구성원
  - jtjun7132@gmail.com

  - 제출 기한 : 16:00
  - 제출해주신 파일을 발표, 시연합니다.