2019 – 1 자료구조 과제

과 목 명 : 자료구조
과 제 명 : 계산기
사용언어 : 파이썬(jupyter notebook)
학　　과 : 심리학과
학　　번 : 20141963
이　　름 : 박승완
제 작 일 : 20190407
담당교수 : 김승태교수님

- 문제 내용 : 계산식을 입력하면 결과를 돌려주는 계산기 프로그램을 작성하라.
  - 사용되는 연산자는 +,-,*,/,%,( ), ** 이다.
  - 수는 1000 이하의 정수만 사용한다. / 는 몫을 구하는 연산자이다.
  - ** 는 제곱을 의미하며 3**5는 3^5이다. **의 우선순위는 ( )보다 느리고 * 보다 빠르다.
  - 모든 코드는 본인이 작성해야 하며, 외부 모듈이나 남이 작성한 코드를 넣어서는 안 된다.
  - 잘못된 수식이라면 어디에 어떤 문제가 있는 것인지 알려줘야 한다.
- 실행 예)
  2+3*4**2-1 = 49
- 해결 방안 :

총 세개의 메서드를 사용한다.
(errorDetection, infixToPostfix, claPostfix)

전체적인 흐름.
1. errorDetection :
User로 부터 input 받아서 errorDetection으로 넘겨주자!
error가 있으면 출력 후 강제종료
없으면 step 2로!

2. get a input expression :
인풋 받은 것이 오류가 없으므로 infixToPostfix 메서드로 넘겨주는 단계 (메인에서 실행)

3. change infix to postfix :
infixToPostfix 메서드에서 진행
피연산자를 만나면 그대로 postfix에 저장
연산자를 만나면 스택에 저장. stack (스택 Top보다 우선 순위가 낮은 연산자가 나오면 그때 postfix에 저장)
(왼쪽 괄호는 우선 순위가 가장 낮은 연산자)
(오른쪽 괄호가 나오면 스택에서 왼쪽 괄호 위에 쌓여있는 모든 연산자를 postfix에 저장)
끝나면 postfix를 return

4. 메인에서 calPostfix로 postfix를 전달

5. calPostfix :
(숫자처리, 연산자 **처리)
calPostfix 메서드에서 계산 진행 후 결과값을 return

6. 메인에서 결과값 출력

- 메서드별 세부 설명 :

errorDetection :
6개의 오류를 잡아낼 수 있다.
오류1 : 연산자로 시작하는 경우
오류2 : 3+4*(2+2**2))-3 와 같이 ")" 가 더 있는 경우
오류3 : 3+4*(2+(2**2-3) 와 같이 ")"가 부족 한 경우
오류4 : 3+4+(2+(2**+2-3)) 와 같이 ** 뒤에 연산자가 오는 경우
오류5 : 3+4+(2+(2*+2-3)) 와 같이 * 뒤에 연산자가 오는 경우
오류6 : 3+4+(2+(2-+2-3)) **와 * 이외의 연산자 뒤에 연산자가 오는 경우
추가적인 오류인 1000초과의 수를 입력하는 경우는 (오류0) infixToPostfix 메서드에서 처리)

인덱스를 이용하여 잘못된 부분을 표시해준다.

infixToPostfix :
우선순위 처리는 priority라는 딕셔너리를 만들어서 각 연산자에 대응하는 우선 순위 숫자 값을 부여하였다. 각 item이 연산자이면 스택의 top의 우선순위와 비교한다.

전체적인 설명으로는 Items 라는 input 값을 한 글자씩 처음부터 끝까지 For문을 돌린다.
Enumerate를 사용하면 처음에 넣었던 input값이 후에 변경되더라도 처음에 들어간 값으로 계속 element가 출력된다는 점을 이용하여 for 문 내에서 Items라는 input값을 변경하고 (10의 자리 이상의 숫자는 가장 큰 수를 제외하고는 !로 바꾸어준다 (예시 : 123 -> 1!!)) 이를 for문에서 출력되는 item과 비교하며 다르면 continue.
Items가 바뀌기에 items_copy를 만들어서 결과값리스트 (postfix)에 저장할 때에는 손상되지 않은 값을 전달한다.

10의자리 이상 의 수 처리는 앞서 설명한대로 10의 자리 이상의 숫자는 가장 큰 수를 제외하고는 !로 바꾸어준다 (예시 : 123 -> 1!!)

위 세가지 설명은 앞선 해결방안의 3번 알고리즘에 대한 추가설명이다.

calPostfix :
결과값리스트(postfix)를 하나씩 읽어가며 피연산자는 무조건 스택에 push한다.
연산자를 읽을 시에 스택에 저장된 두 개의 피연산자를 pop하여 연산 후 스택에 push한다.
계산이 완료되면 결과값을 return한다.

추가정보 : jupyter notebook에서 코딩하였습니다. 확장자도 .ipynb입니다.

- 결과 표시 :

Test case

```
정상1 : (11+3)*2**3-12 = 100
정상2 : 44+22**(231-39*33)-33%2**4 = 43
정상3 : 44+(231-39)*33-6/2+33 = 6410
정상4 : 7-3-4*2-43 = -47
정상5 : 2+(3*4)**2-12 = 134

오류0 : 1010+1
오류1 : +3+4*2
오류2 : 3+4*(2+2**2))-3
오류3 : 3+4*(2+(2**2-3)
오류4 : 3+4*(2+(2**+2-3))
오류5 : 3+4+(2+(2*+2-3))
오류6 : 3+4+(2+(2-+2-3))
```

정상1 :

```
In [266]: main()

-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
(11+3)*2**3-12
계산식:  (11+3)*2**3-12
결과:  100
```

정상2

```
In [267]: main()

-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
44+22**(231-39*33)-33%2**4
계산식:  44+22**(231-39*33)-33%2**4
결과:  43
```

정상3

```
In [268]: main()

-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
44+(231-39)*33-6/2+33
계산식:  44+(231-39)*33-6/2+33
결과:  6410
```

정상4

```
In [269]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
7-3-4*2-43
계산식:  7-3-4*2-43
결과:  -47
```

정상5

```
In [270]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
2+(3*4)**2-12
계산식:  2+(3*4)**2-12
결과:  134
```

오류0

```
In [271]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
1010+1
오류0
1010(!)+1 이 위치에 오류가 있습니다. 입력값 중 1000이 넘는 수가 있습니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류1

```
In [272]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
+3+4*2
오류1
+(!)3+4*2 이 위치에 오류가 있습니다. 연산자로 시작합니다

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류2

```
In [273]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
3+4*(2+2**2))-3
오류2
3+4*(2+2**2))(!)-3이 위치에 오류가 있습니다. )가 하나 더 있습니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류3

```
In [274]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
3+4*(2+(2**2-3)
오류3
3+4*(2+(2**2-3)(!)이 위치에 오류가 있습니다. )가 부족합니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류4

```
In [275]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
3+4*(2+(2***+2-3))
오류4
3+4*(2+(2***+(!)2-3))이 위치에 오류가 있습니다. 연산자가 중복되어 나옵니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류5

```
In [276]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
3+4+(2+(2*+2-3))
오류5
3+4+(2+(2*+(!)2-3))이 위치에 오류가 있습니다. 연산자가 중복되어 나옵니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```

오류6

```
In [277]: main()
```

```
-------계산기-------
규칙 : 1000이하의 정수
규칙 : 사용가능한 연산자 -> **, *, %, /, +, -, (, )
규칙 : 계산을 원하는 식을 다음의 방식으로 입력하시오 -> 예) 3+5*(50-3**2)-1
3+4+(2+(2-+2-3))
오류6
3+4+(2+(2-+(!)2-3))이 위치에 오류가 있습니다. 연산자가 중복되어 나옵니다.

An exception has occurred, use %tb to see the full traceback.

SystemExit: error
```