**프로그래밍 언어개론 과제 보고서**

Item02

Cute17 Project

01분반\_13조

201204441 김수현

201302363 권오현

201102448 사명기

1. **구현 방식**
2. **Define**

* Define의 구현에 앞서 Define의 정의 및 이해가 필요하였다. Define이란 (define a 1)과 같이 사용되며, 파이썬에서 딕셔너리 자료형 구조를 사용하여 a라는 key값이 들어오면 뒤에 오는 1은 value로써 저장을 하게 된다. 즉, 딕셔너리 구조에 맞춰 {key:value, key2:value2}형태로 저장되는 것이다.
* 위의 define을 통해 저장해야할 key와 value를 저장할 전역 딕셔너리 매트릭스가 필요하였으며, insertDicM이라는 딕셔너리 매트릭스를 만들었다. (이하 insertDicM)
* 해당 key, value를 받아서 저장할 기능을 수행할 함수가 필요하였다. 이를 위하여 def insert\_table를 만들었으며, 해당 부분에서는 l\_node.value(ID값)와 run\_expr(r\_node)(INT 등 value로서 저장할 값)의 값을 각각 key와 value로 받아서 insertDicM에 저장하게 하였다.
* insertDicM에 저장한 값을 가져다가 run\_func(op\_code\_node)에 정의된 함수들이 가져다가 사용을 해야하고, 혹, Key값을 입력하면 insertDicM에 존재하면 해당 value를 연산시에 사용해야하기 때문에 lookup\_table(id)이라는 함수를 만들어주었다.
* 앞서 위에서 연산시, 해당 값이 define을 통해 저장된 key:value쌍이 있는지 lookup\_table에서 한번 찾아보고자 run\_expr내의 ID타입시에 해당 lookup\_table에 들어가 해당 부분의 과정을 거치게 하였다.
* 해당 lookup\_table은 id값(key값)을 받아와서 insertDicM[id]을 하여 해당 key값을 딕셔너리에서 찾아서 key에 대응되는 value값을 return하게 하였고, 없으면 그대로 넣은 node.value값을 반환하게 구성하였다.
* 사용자가 (define a 1)을 한 후 a 를 쳤을 시 해당 key인 a가 insertDicM에 존재한다면 해당 값을 출력시켜야 하기 때문에 print\_node부분에 ID타입 INT타입 노드가 들어오는 부분에서 조건을 넣어 insertDicM에 해당 key(“a”)가 있다면 insertDicM에서 대응되는 value값을 반환하게 코드를 수정하였다.

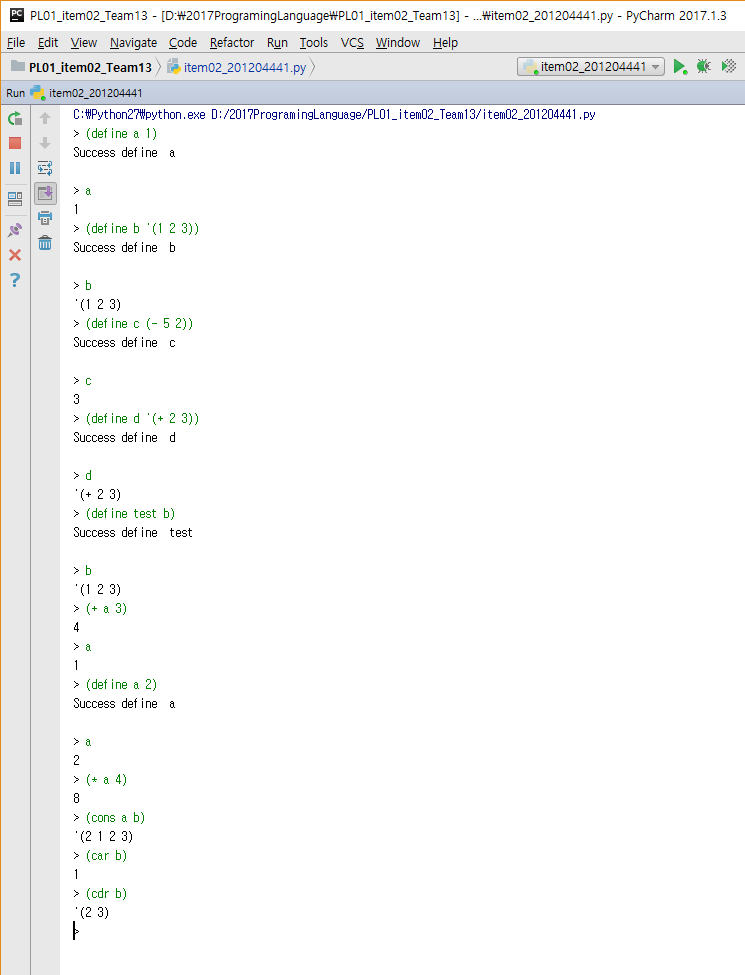
1. **LAMBDA**

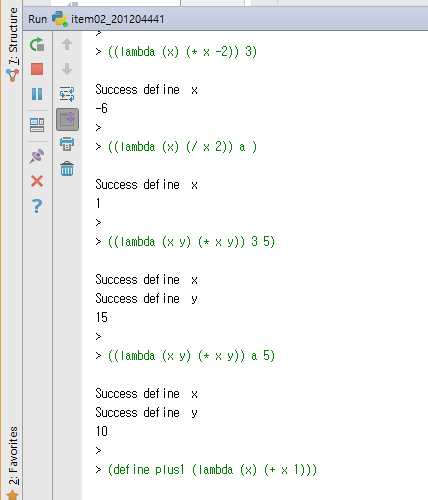
* 람다의 경우 식을 대응시켜 저장하는 형태이고 ((LAMBDA (x) (+ x 1)) 2) 일시 해당 2의 값을 x에 대입하여 (+ 2 1)을 수행하게 끔 하는 형태이다.
* 또한 함수형 언어에서 define과 lambda를 이용하여 해당 식을 저장하고 이를 불러다 사용하는 형태로 되어있다. 이를 구현하기 위한 LAMBDA가 필요한 것이다.
* 이를 위하여 앞서 LAMBDA를 정의해줄 필요가 있다. 허나, LAMBDA라는 함수 자체가 이미 Python에 존재하기 때문에 해당 부분에 있어서 run\_lambda로 대체하여 이를 table에 넣어 인식하게 하였다.
* Run\_lambda에서는 변수를 3개를 사용하였다. 첫째로는 람다식의 매개변수(x)를 받게 그후 람다 식의 실행부에 속하는 (+ x 1) 의 node를 받고 마지막으로 해당 람다식의 매개변수에 바인딩 시켜줄 실인자를 넣는 부분으로 3개의 변수를 구성하였다.
* run\_lambda에서는 우선 ((lambda (x) (\* x -2)) 3), ((lambda (x y) (\* x y)) 3 5) 이와 같이 매개변수를 1개 혹은 1개 이상으로 받을 경우를 나누어 계산을 수행한다.
* 실행부분 (\* x 1)과 같은 수행부가 2개 이상 나오는 (define plus2 (lambda (x) (+ (plus1 x) 1)))것과 같은 식을 수행하기 위하여 수행부가 더 없을 때 까지 while문을 만들어 끝날 때 까지 지속적으로 수행하게 된다. 이와 같은 수행부를 모두 수행하면 수행 한 결과를 result에 저장하여 반환하게 하여 최종 람다식을 수행한 결과를 볼 수 있게 된다.
* 람다부에서 실행시 필요한 위의 변수들에 넣을 값들과 또 추가적으로 필요한 람다식의 수행을 위하여 run\_expr(root\_node)부분의 LIST부분을 수정하였다.
* 해당 리스트 부분에서는 ((lambda (x) (+x 1)) 3) 와 같이 2겹으로 쌓여 있는 경우 LIST 속에 value인 ((lambda (x) (+x 1)) 3 이렇게 만들어 줘서 위의 람다에서 위에 설명한 단계를 수행하며 람다를 할 수 있게 진행을 만들어 줬다.
* (define plus1 (lambda (x) (+ x 1))) 이와 같은 식이 왔을 시에는 define에 의거 plus1이 딕셔너리 insertDicM에 저장되는데, 이 후 (plus1 3)을 수행시 그 뒷부분인 else부분에서 수행을 진행하게 되며, 해당 부분이 insertDic에 들어있는지 확인을 하게 된다. Plus1의 경우 insertDicM안에 정의가 되어 있기에 그 부분을 내부에서 수행을 진행하며, 매개변수 x에 바인딩 해줄 3을 같이 엮어서 (lambda (x) (+x 1)) 3의 노드를 root\_node라는 변수에 넣어 이를 run\_list(root\_node)를 하여 앞서 말한 lambda에 정의된 대로 수행하게 만들어준다.
* 이와 같이 람다를 구성하였고, 실제 테스트를 20개 까지 모두 돌렸으나, 17번 20번은 원래 의도된대로 돌아가지 않았다.
* 20번의 경우 cube까지 모두 작동은 하나 내부에 임시로 정의했던 부분인 sqrt가 다시 불러보면 딕셔너리 테이블에서 불러와 짐으로, 이 부분은 추가적으로 추후 개선의 필요성이 있겠다.

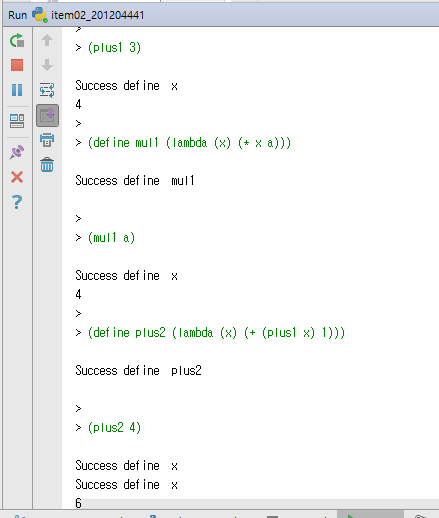
1. **역할 분담 및 느낀점**

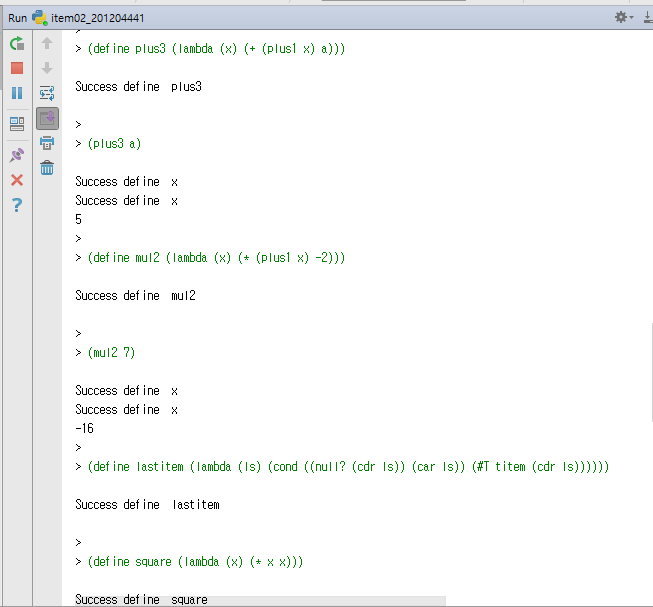
|  |  |
| --- | --- |
| 김 수 현 | Define의 전체적인 부분을 코딩하였고, 해당 부분에 있어서 오류가 발생하여 팀원들의 피드백 및 refactory를 거쳐서 오류를 완전히 수정하고, test code의 define에 해당하는 7번까지 모두 결과를 뽑아낼 수 있었다. refactory과정에서 존재하였던 기타 수정할 코드 및 오류사항들을 잡았다. 추가적으로 팀의 Github사용을 모두 총괄하여 진행하였다. 팀의 github관리와 필요시 교육, 보고서 등을 담당하였다. 이번에 파이썬을 통하여 interpreter를 구현함으로서, 코딩에 약간은 자신감이 붙었지만 아직 많이 부족함을 느낀다. 추가적으로 git에 대한 사용법은 기본사용법은 모모 익혔지만, 팀원들에게 모두 완벽히 이해하게 전파하지 못한 점은 아쉬웠다. LAMDA의 구현이 확실하게 완성이 되었으면 하였지만 이 부분은 여전히 아쉬움으로 남는다. 서로 코딩을 하면서 리뷰하고 장단점을 파악하여, 고쳐나감으로서 스스로 얻을 수 없는 값진 경험을 한 것이 좋았다. 추후 코드 중복이 많은 run\_func부분과 구현 못한 LAMBDA를 다시 한번 도전해보면 어떨까 한다. |
| 권 오 현 | Lambda 함수 구현, 처음 lambda를 만들어 실행할 땐 a 테스트를 완료하면 b 테스트를 통과하지 못하는 시행착오를 겪으면서 node의 연산이 어떻게 처리되는지 하나하나 debug해 보았고, 이를 통해  구현할 interpreter의 연산의 메커니즘을 좀 더 이해할 수 있었습니다. 원하는 value값에 접근하기 위해 여러 접근방법을 시도했지만,  깔끔하고 보기좋은 코드를 사용해 구현하진 못해서 많이 아쉽습니다. (Value.value)  또, 이번 과제는 git을 사용해 진행되었는데 이론으로만 배웠던 git을 실제 과제에 적용하는 것은 처음이라 어색하고 모르는 부분도 많았지만, 팀원의 git 교육을 통해 조금 더 git을 이해하는 좋은 계기가 되었습니다 |
| 사 명 기 | 이런 걸 한다고 생각하지 못하고 수강했는데 막상 하려니 생소한 걸 하는 것과 새로운 언어인 파이썬이라는 언어에 익숙해지기가 여간 쉽지 않았다. item2를 구현할 땐 파이썬의 기본 구조에 대해서도 찾아봐야 했다. 이번 프로젝트를 통해 프로그래밍 언어의 구조와 처리에 대해 조금 더 이해할 수 있었다. item2을 진행할 때 팀원에게 많은 도움을 받았다. 과제를 하면서 다 짜진 코드를 보면 대부분 몇 줄 되지도 않는데 왜 난 이걸 생각 못했었을까 싶었던 경우도 많아 어이없던 일도 많았다. 하지만 여태껏 다뤄보지 못한 분야의 과목을 들은 것 같아 뭔가 새로운 영역에 대해 알 수 있었던 것 같다. 하나의 인터프리터를 완벽하진 않지만 구현해봤다는 게 신기하다. 더 열심히 못했던 부분들이 있어 아쉬움도 많이 남는다 |

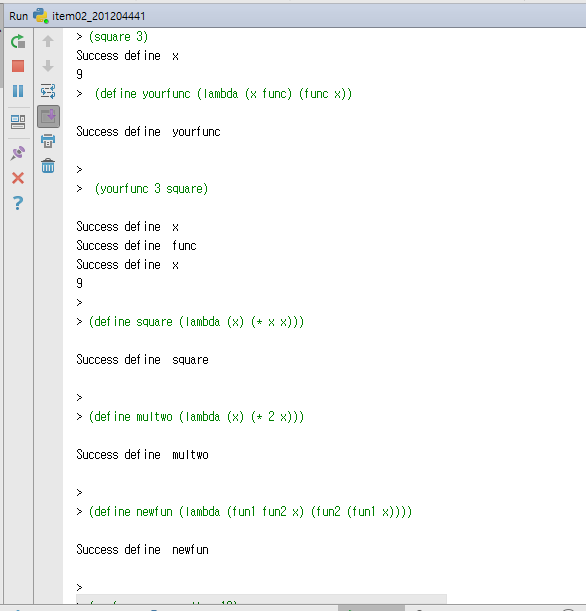
1. **결과 화면**

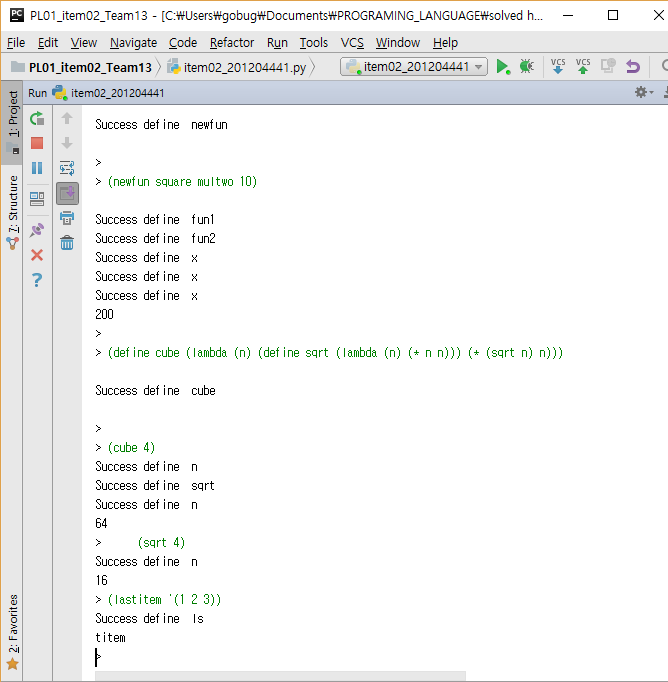


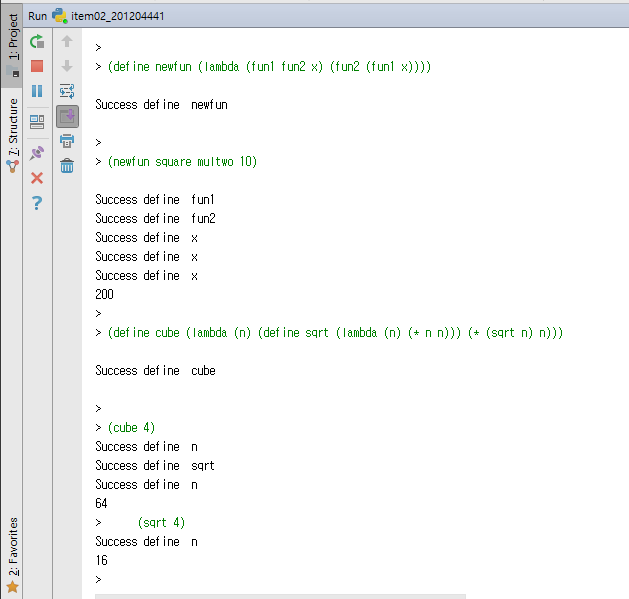












1. **Git 주소**

https://github.com/skyfall13/PL01\_item02\_Team13.git