AIP

HW 07 – REPORT

소속 : 정보컴퓨터공학부 컴퓨터공학전공

학번 : 202355570

이름 : 이유진

1. 서론

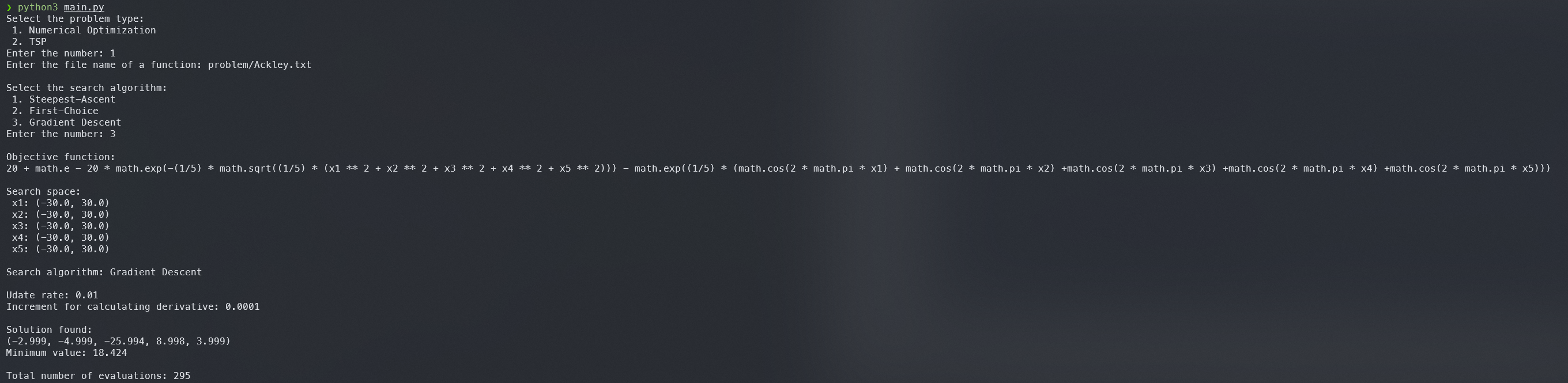
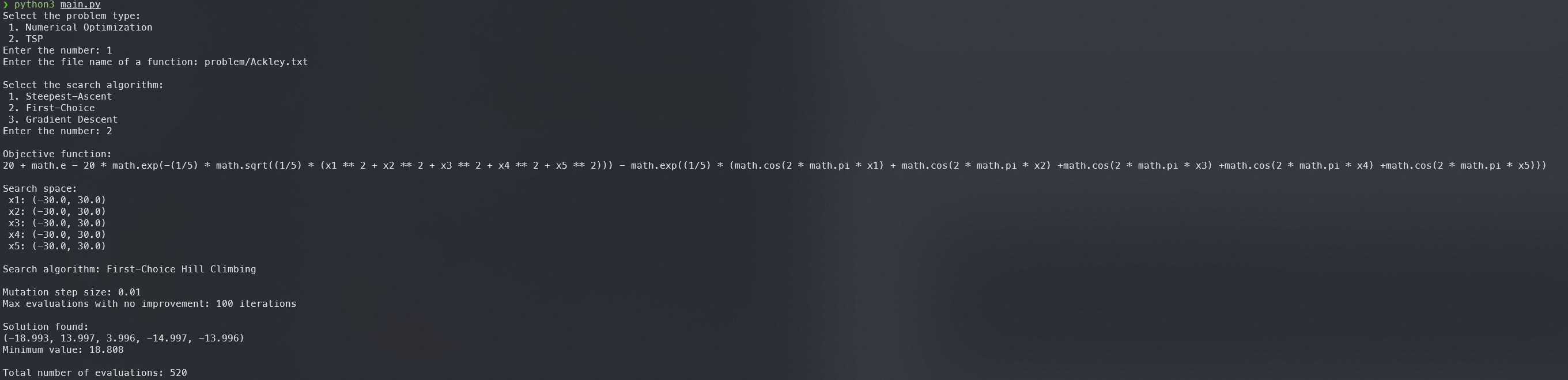
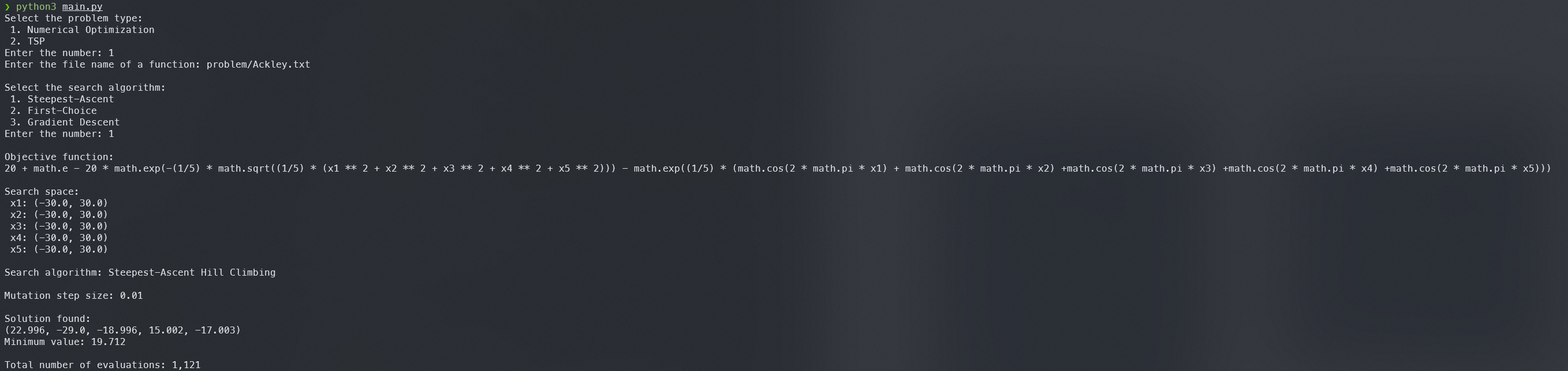
이전 과제에서 만든 class들을 하나의 프로그램으로 통합하여 한 프로그램 (main.py) 만을 실행시켜도 여러가지 문제를 여러 방식으로 해결할 수 있도록 구현해보자.

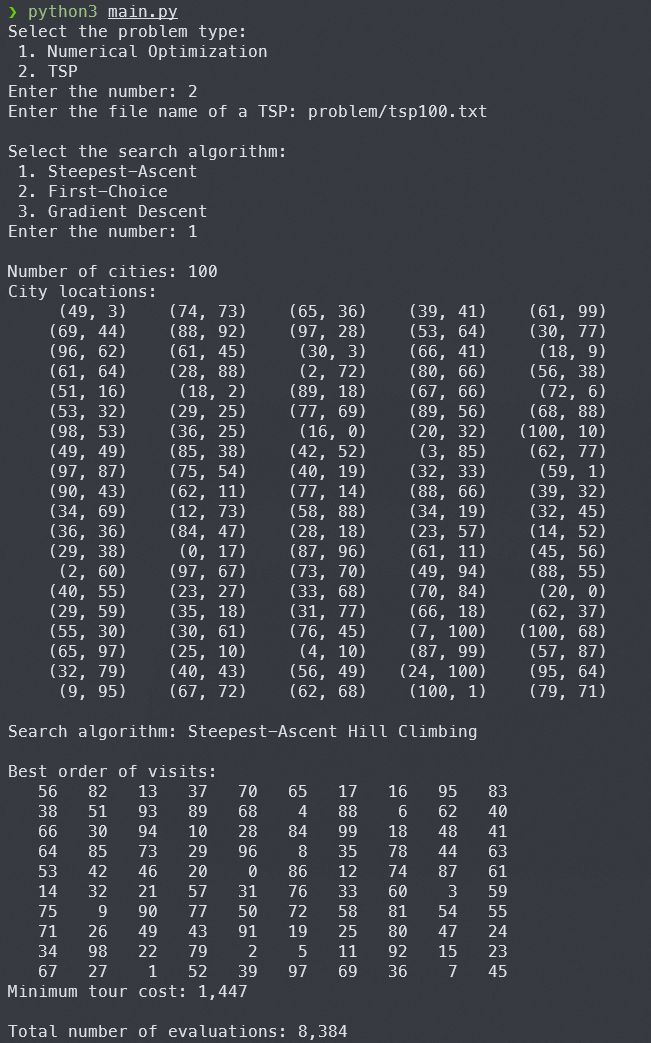
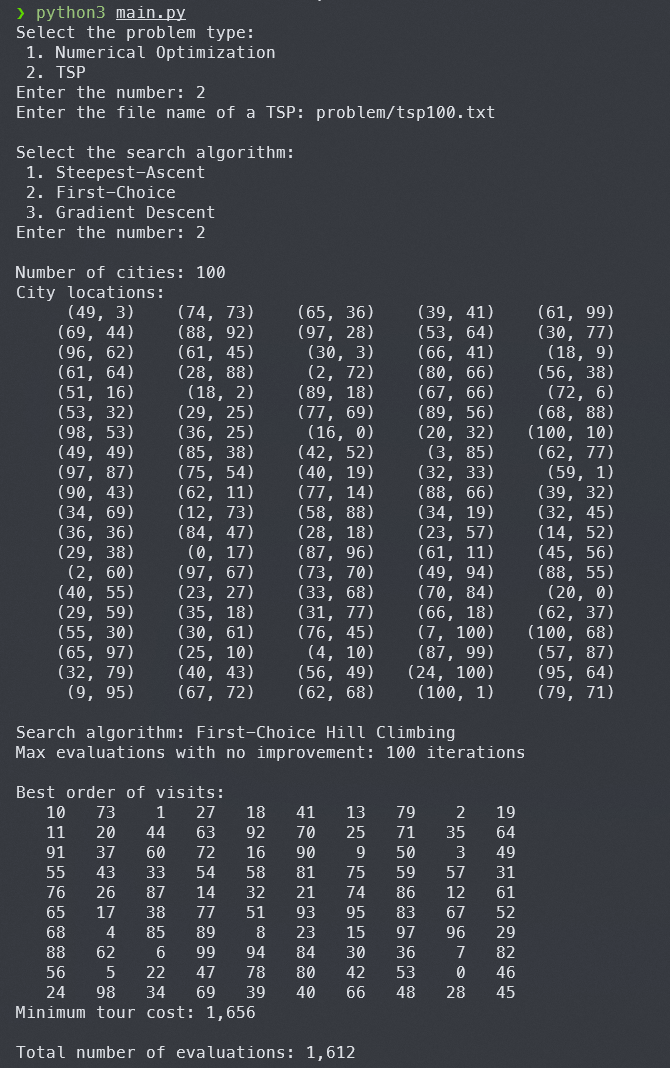
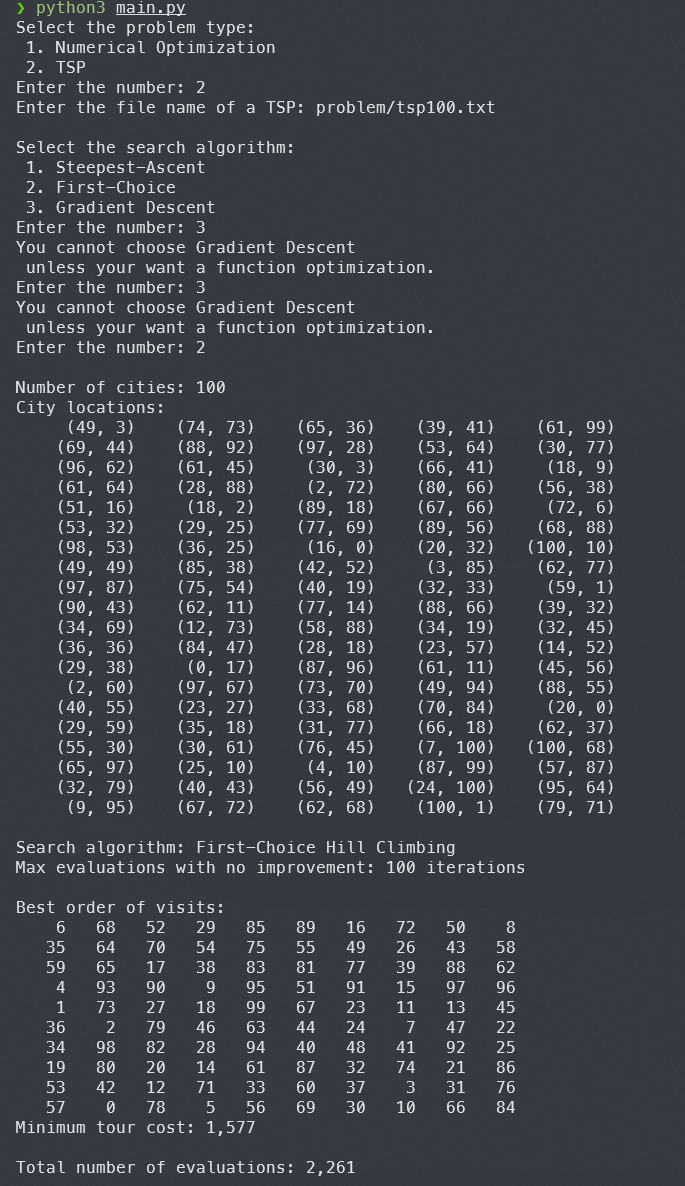
Setup라는 슈퍼클래스를 구현하여 HillClimbing 클래스와 Problem 클래스에서 모두 필요한 정보를 공유할 수 있도록 만들어보자.

SteepestAcent, FirstChoice, GradientDescent 세가지 검색 알고리즘을 통합시켜 HillClimbing class를 구현해보자.

2. 본론

1. 먼저 Setup.py에서 Setup class를 구현하여 HillClimbing 클래스와 Problem 클래스에서 모두 필요한 정보를 공유할 수 있도록 self.\_delta, self.\_alpha, self.\_dx를 초기화할 수 있또록 만든다.
2. Problem.py 에서 `from setup import Setup`로 setup 클래스를 import해주고, Problem(Setup)으로 해당 클래스를 상속하도록 만들어준다. 슈퍼클래스를 호출하여 delta, alpha, dx를 초기화해주고 기존의 따로 초기화하던 부분을 없애준다.
3. Optimizer.py 에서 Setup부분을 import하고 HillClimbing class를 만들어 Setup을 상속받도록 만들어준다. \_\_init\_\_에서 Setup에서 정의된 delta, alpha,dx에 접근할 수 있도록 슈퍼클래스의 생성자를 호출해주고 문제 Type을 구분하기위한 pType을 정의해준다.
4. pType을 초기화할 수 있는 setVariables 메서드와 공통적으로 출력되는 부분을 displaySetting 메서드를 만들어준다.
5. FristChoice, SteepestAscent. GradientDescent 서브클래스를 만들어 HillClimbing class를 상속받게 만들어주고 저번 과제에서만든 displaySetting을 가져와 각각 따로 구현해준다. 공통으로 출력되는 부분은 super().displaySetting()를 호출하여 출력해준다. 슈퍼클래스에서 선언만 해놓은 run부분을 만들어줌. (이전 과제 만든 FirstChoice(), SteepestAscent(), GradientDescent() 부분의 내용을 가져와서 구현한다.
6. SteepestAscent의 경우 bestOf를 추가적으로 메서드로 만들어 구현해준다. 파라미터에 self를 추가해주고, run 메서드 부분에 self.bestOf(neighbors,p)를 호출하도록 만들어주었다.
7. GradientDescent의 경우 출력방식이 약간달라 self.\_alpha, self.\_dx로 따로 출력해주었따.
8. Main.py 로 여러가지 문제를 여러가지 방식으로 풀 수 있도록 문제타입과 알고리즘을 선택할 수 있는 함수를 만들어 준다. 만약 problemType이 ==2 즉, TSP인 경우에는 Gradient Descent 를 사용할 수 없으므로 알고리즘 Type로 Gradient Descent 선택시 예외처리를 해줘야함
9. Invalid 함수를 구현하여 aType==2 && pType==3 인 경우 True를 리턴하게 하여 올바른 값이 나올때까지 입력을 받게 만들어준다.



3. 결론

Problem라는 슈퍼클래스를 구현하여 Numeric과 Tsp 문제를 서브클래스로 구현하고 Hill Climbing라는 슈퍼클래스를 구현하여 SteepestAcent, FirstChoice, GradientDescent 세가지 검색 알고리즘을 통합시킬 수 있다.

이와 같은 코드 최적화를 통해 하나의 파일로 여러가지 문제를 여러가지 방식으로 해결해 나갈 수 있고 하나의 파일로 서로 다른 실행결과를 더 쉽게 분석할 수 있다.

Numeric (Ackley.txt) 문제에서

1. Steepest-Ascent Hill Climbing
   1. Solution found: (22.996, -29.0, -18.996, 15.002, -17.003)
   2. Minimum value: 19.712
   3. Total number of evaluations: 1,121

모든 이웃을 탐색하며 최적의 방향을 찾아 나가는 알고리즘이므로 다른 알고리즘 방식들 중에 가장 많은 평가 횟수가 나왔고 가장 속도가 느림을 알 수 있다.

1. Frist-Choice Hill Climbing
   1. Solution found: (-18.993, 13.997, 3.996, -14.997, -13.996)
   2. Minimum value: 18.808
   3. Total number of evaluations: 520

Successor을 생성하다가 자기 자신(현 상태)보다 좋은 것을 발견하면 바로 이동하는 알고리즘이므로 Steepest-Ascent Hill Climbing 보다는 더 적은 평가 횟수가 나왔고 이 보다는 속도가 빠름을 알 수 있다.

1. Gradient Descent
   1. Solution found: (-2.999, -4.999, -25.994, 8.998, 3.999)
   2. Minimum value: 18.424
   3. Total number of evaluations: 295

기울기를 이용하여 최적의 값을 찾아 나가는 알고리즘으로 세가지 방식 중에서 가장 평가 횟수가 적게 나타났고 가장 속도가 빠른 것을 알 수 있다.

Numeric 문제(Ackley.txt) 에 있어서 Gradient Descent > Frist-Choice > Steepest-Ascent 순서로 최적화 성능이 좋다고 볼 수 있다.

Tsp (tsp100.txt) 문제에서

1. Steepest-Ascent Hill Climbing
   1. Minimum tour cost: 1447
   2. Total number of evaluations: 8384

Steepest-Ascent 의 경우는 빠른 탐색보다는 최적의 방향을 찾는데에 집중하고 있으므로 평가횟수가 높게 나타나 시간이 오래 걸린다.

1. First-Choice Hill Climbing
   1. Minimum tour cost: 1656
   2. Total number of evaluations: 1612

평가횟수가 Steepest-Ascent 보다 낮게 나타나 빠른 것을 알 수 있고 비용측면에서는 Steepest-Ascent 보다 더 높게 나타나 최적의 해에 덜 가까운 결과를 도출하는 것을 볼 수 있다.

1. Gradient Descent

기울기를 이용한 알고리즘이므로 tsp문제에서는 적절하지 않다.

Tsp 문제 (tsp100.txt)에 있어서 속도 측면에서 First-Choice > Steepest-Ascent 순서로 성능이 좋다고 볼 수 있다.