AI 프로그래밍 059분반

HW9 보고서

2020년 11월 18일

정보컴퓨터공학과

201824481 박지우

목차

1. Setup.py

```python
class Setup():
    def __init__(self):
        self.delta = 0.01
        self.updateRate = 0.01
        self.dx = 10 ** (-4)

    def getDelta(self):
        return self.delta

    def getUpdateRate(self):
        return self.updateRate

    def getDx(self):
        return self.dx
```

Setup.py의 파일에 Setup 클래스를 선언해주었다. Problem과 HillClimbing에서 모두 사용되는 delta, updateRate, dx값을 변수로 선언했으며 각 변수의 값을 리턴하는 함수를 정의했다.

2. Problem.py

2-1. Class Problem

```python
import random
import setup
import math


class Problem(setup.Setup):
    def __init__(self):
        super().__init__()
        self.solution = None
        self.minimum = None
        self.NumEval = 0
```

```python
    def saveResult(self, solution, minimum):
        self.solution = solution
        self.minimum = minimum

    def report(self):
        print()
        print("Total number of evaluations: {0:,}".format(self.NumEval))
```

Problem.py파일은 지난 과제에서 사용된 코드와 매우 유사하다. 슈퍼클래스로 Setup을 사용하게 되었으므로 import하고 Problem의 슈퍼클래스가 Setup임을 나타낸다. 기존의

delta, limitStuck 등의 변수는 제거해준다. 결과값을 저장해주는 saveResult 함수와 Numeric, Tsp 모두 출력하는 결과를 report 함수로 출력한다.

2-2. Class Numeric

```
22  class Numeric(Problem):
23      def __init__(self):
24          super().__init__()
25          self.domain = None
26          self.expression = None
```

지난 과제에 사용된 Numeric 클래스의 updateRate 변수를 제거해준다. updateRate변수는 Setup 클래스에 정의된다.

```
28      def setVariables(self, fileName):
29          varNames = []
30          low = []
31          up = []
32          infile = open(fileName, 'r')
33          lines = infile.readlines()
34          infile.close()
35
36          for i in range(len(lines)):
37              lines[i] = lines[i].rstrip()
38          self.expression = lines[0]
39
40          for i in range(1, len(lines)):
41              data = lines[i].split(',')
42              varNames.append(data[0])
43              low.append(float(data[1]))
44              up.append(float(data[2]))
45          self.domain = [varNames, low, up]
```

createProblem 함수의 이름을 setVariables로 바꾸었다. 또한 지난 주 코드에서는 파일 이름을 setVariables 함수 내에서 입력 받았지만, 이번 과제는 main에서 입력하므로 filename 인자를 통해 파일 이름을 전달받는 형식으로 바꾸었다.

```
47      def randomInit(self):
48          init = []
49          for i in range(0, 5):
50              data = random.uniform(self.domain[1][i], self.domain[2][i])
51              init.append(data)
52          return init
53
54      def evaluate(self, current):
55          self.NumEval += 1
56          expr = self.expression
57          varNames = self.domain[0]
58          for i in range(len(varNames)):
59              assignment = varNames[i] + '=' + str(current[i])
60              exec(assignment)
61          return eval(expr)
```

```
63      def mutate(self, current, i):
64          curCopy = current[:]
65          l = self.domain[1][i]
66          u = self.domain[2][i]
67          if l <= (curCopy[i] + self.delta) <= u:
68              curCopy[i] += self.delta
69          return curCopy
```

```
71      def describe(self):
72          print()
73          print("Objective function:")
74          print(self.expression)
75          print("Search space:")
76          varNames = self.domain[0]
77          low = self.domain[1]
78          up = self.domain[2]
79          for i in range(len(low)):
80              print(" " + varNames[i] + ":", (low[i], up[i]))
```

```
82      def coordinate(self):
83          c = [round(value, 3) for value in self.solution]
84          return tuple(c)
85
86      def report(self):
87          print()
88          print("Solution found:")
89          print(self.coordinate())
90          print("Minimum value: {0:,.3f}".format(self.minimum))
91          super().report()
```

```
    def randomMutant(self, current):
        i = random.randint(0, 4)
        return self.mutate(current, i)

    def mutants(self, current):
        neighbors = []
        for i in range(0, 5):
            neighbors.append(self.mutate(current, i))
        return neighbors
```

```
    def isLegal(self, x):
        low = self.domain[1]
        up = self.domain[2]
        flag = True
        for i in range(len(low)):
            if x[i] < low[i] or up[i] < x[i]:
                flag = False
                break
        return flag
```

다음은 지난 과제의 코드와 같다. 다만 describeProblem의 함수명이 describe로 바뀌었으며, displayResult의 함수명을 report로 바꾸었다. Gradient-Descent 구현을 Search Algorithm (C) 강의에서 교수님께서 알려주신 방향으로 바꾸어 isLegal 함수를 추가하였다.

2-3. Class Tsp

```
class Tsp(Problem):
    def __init__(self):
        super().__init__()
        self.numCities = 0
        self.locations = []
        self.table = []

    def setVariables(self, fileName):
        infile = open(fileName, 'r')
        self.numCities = int(infile.readline())
        line = infile.readline()
        while line != '':
            self.locations.append(eval(line))
            line = infile.readline()
        infile.close()
        self.calcDistanceTable()
```

Tsp 클래스에서도 createProblem의 함수명을 setVariables로 바꾸었으며, 파일 이름을 인자로 받아오는 형태로 바꾸었다.

```
131     def calcDistanceTable(self):
132         for i in range(self.numCities):
133             data = []
134             for j in range(self.numCities):
135                 data.append(math.sqrt(math.pow((self.locations[i][0] - self.locations[j][0]), 2)
136                                       + math.pow((self.locations[i][1] - self.locations[j][1]), 2)))
137             self.table.append(data)
```

```
139     def randomInit(self):
140         init = list(range(self.numCities))
141         random.shuffle(init)
142         return init
143 
144     def evaluate(self, current):
145         self.NumEval += 1
146         cost = 0
147         for i in range(self.numCities - 1):
148             a = current.index(i)
149             b = current.index(i + 1)
150             cost += self.table[a][b]
151         return cost
```

```
153     def inversion(self, current, i, j):
154         curCopy = current[:]
155         while i < j:
156             curCopy[i], curCopy[j] = curCopy[j], curCopy[i]
157             i += 1
158             j -= 1
159         return curCopy
```

```
161     def describe(self):
162         print()
163         print("Number of cities:", self.numCities)
164         print("City locations:")
165         for i in range(self.numCities):
166             print("{0:>12}".format(str(self.locations[i])), end='')
167             if i % 5 == 4:
168                 print()
```

```
170     def report(self):
171         print()
172         print("Best order of visits:")
173         self.tenPerRow()
174         print("Minimum tour cost: {0:,}".format(round(self.minimum)))
175         print()
176         print("Total number of evaluations: {0:,}".format(self.NumEval))
177 
178     def tenPerRow(self):
179         for i in range(len(self.solution)):
180             print("{0:>5}".format(self.solution[i]), end='')
181             if i % 10 == 9:
182                 print()
```

```
184     def randomMutant(self, current):
185         while True:
186             i, j = sorted([random.randrange(self.numCities)
187                            for _ in range(2)])
188             if i < j:
189                 curCopy = self.inversion(current, i, j)
190                 break
191         return curCopy
```

```
193     def mutants(self, current):
194         neighbors = []
195         count = 0
196         triedPairs = []
197         while count <= self.numCities:
198             i, j = sorted([random.randrange(self.numCities) for _ in range(2)])
199             if i < j and [i, j] not in triedPairs:
200                 triedPairs.append([i, j])
201                 curCopy = self.inversion(current, i, j)
202                 count += 1
203                 neighbors.append(curCopy)
204         return neighbors
```

다음은 지난 과제의 코드와 같다. 다만 describeProblem의 함수명이 describe로 바뀌었으며, displayResult의 함수명을 report로 바꾸었다.

3. Optimizer.py

3-1. Class HillClimbing

```
1   import setup
2
3
4   class HillClimbing(setup.Setup):
5       def __init__(self):
6           super().__init__()
7           self.limitStuck = 100
```

Setup을 슈퍼클래스로 하는 HillClimbing 클래스를 선언한다. 사용되는 변수는 기존 Problem에 정의되었던 limitStuck이다.

3-2. Class SteepestAscent

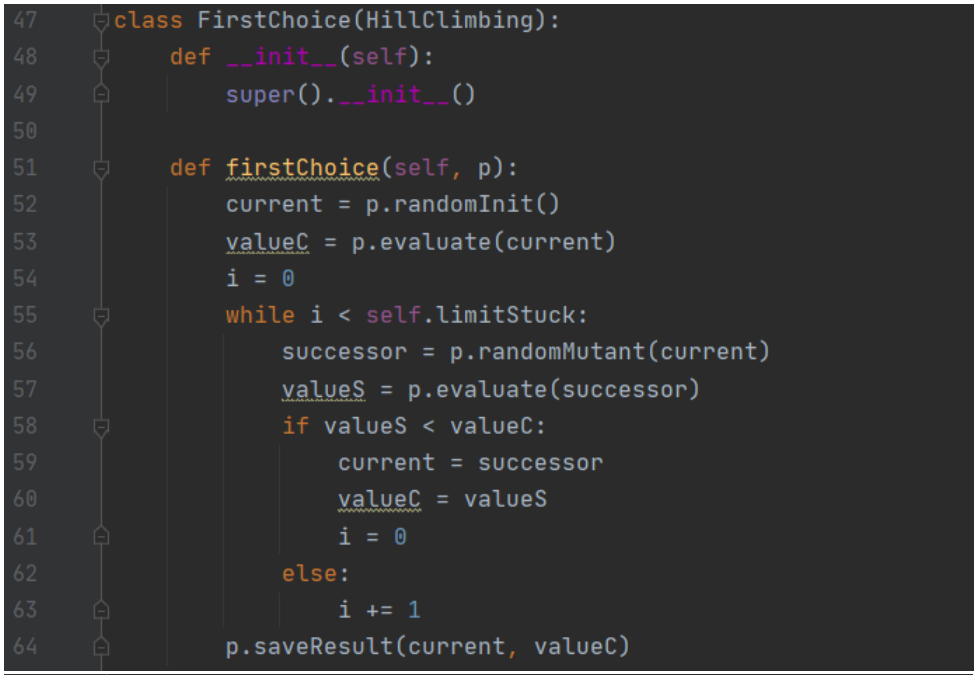
```
class SteepestAscent(HillClimbing):
    def __init__(self):
        super().__init__()

    def steepestAscent(self, p):
        current = p.randomInit()
        valueC = p.evaluate(current)
        while True:
            neighbors = p.mutants(current)
            successor, valueS = self.bestOf(neighbors, p)
            if valueS >= valueC:
                break
            else:
                current = successor
                valueC = valueS
        p.saveResult(current, valueC)
```

```
    def bestOf(self, neighbors, p):
        for i in range(len(neighbors)):
            value = p.evaluate(neighbors[i])
            if i == 0:
                bestValue = value
                best = neighbors[i]
            elif value < bestValue:
                bestValue = value
                best = neighbors[i]
        return best, bestValue
```

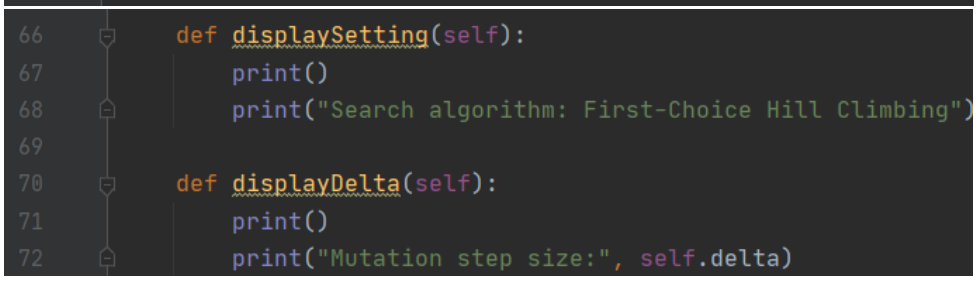
```
    def displaySetting(self):
        print()
        print("Search algorithm: Steepest-Ascent Hill Climbing")

    def displayDelta(self):
        print()
        print("Mutation step size:", self.delta)
```

steepestAscent 함수는 지난 과제 코드 중 Steepest-ascent (n).py 파일의 일부이다. Numeric 뿐만 아니라 Tsp도 동일하다. 또한 Setting을 출력하되, delta값은 Numeric의 경우에만 출력하므로 따로 함수를 선언하였다.

3-3. Class FirstChoice



```python
class FirstChoice(HillClimbing):
    def __init__(self):
        super().__init__()

    def firstChoice(self, p):
        current = p.randomInit()
        valueC = p.evaluate(current)
        i = 0
        while i < self.limitStuck:
            successor = p.randomMutant(current)
            valueS = p.evaluate(successor)
            if valueS < valueC:
                current = successor
                valueC = valueS
                i = 0
            else:
                i += 1
        p.saveResult(current, valueC)
```



```python
    def displaySetting(self):
        print()
        print("Search algorithm: First-Choice Hill Climbing")

    def displayDelta(self):
        print()
        print("Mutation step size:", self.delta)
```

firstChoice 함수는 지난 과제 코드 중 First-choice (n).py 파일의 일부이다. Numeric 뿐만 아니라 Tsp도 동일하다. 또한 Setting을 출력하되, delta값은 Numeric의 경우에만 출력하므로 따로 함수를 선언하였다.

3-4. Class GradientDescent

```
75  class GradientDescent(HillClimbing):
76      def __init__(self):
77          super().__init__()
78
79      def gradientDescent(self, p):
80          current = p.randomInit()
81          valueC = p.evaluate(current)
82          while True:
83              nextP = self.takeStep(current, valueC, p)
84              valueN = p.evaluate(nextP)
85              if valueN >= valueC:
86                  break
87              else:
88                  current = nextP
89                  valueC = valueN
90          p.saveResult(current, valueC)
```

```
92      def takeStep(self, x, v, p):
93          grad = self.gradient(x, v, p)
94          xCopy = x[:]
95          for i in range(len(xCopy)):
96              xCopy[i] = xCopy[i] - self.updateRate * grad[i]
97          if p.isLegal(xCopy):
98              return xCopy
99          else:
00              return x
01
02      def gradient(self, x, v, p):
03          grad = []
04          for i in range(len(x)):
05              xCopyH = x[:]
06              xCopyH[i] += self.dx
07              g = (p.evaluate(xCopyH) - v) / self.dx
08              grad.append(g)
09          return grad
```

```
92      def displaySetting(self):
93          print()
94          print("Search algorithm: Gradient-Descent Hill Climbing")
95
96      def displayUpdateDx(self):
97          print()
98          print("Update rate:", self.updateRate)
99          print("Increment for calculating derivatives:", self.dx)
```

gradientDescent 함수는 지난 과제 코드 중 Gradient-descent (n).py 파일의 일부이다. 또한 Setting을 출력하되, 추가적으로 updateRate와 dx값을 출력하는 함수를 선언하였다.

4. Main.py

```python
import optimizer
import problem

print("Select the problem type:")
print("\t1. Numerical Optimization")
print("\t2. TSP")
type = int(input("Enter the number: "))
file = str(input("Enter the file name of a function: "))

print("Select the search algorithm:")
print("\t1. Steepest-Ascent")
print("\t2. First-Choice")
print("\t3. Gradient Descent")
algorithm = int(input("Enter the number: "))
```

Problem과 optimizer을 import한 후, 과제에 제시되어 있는 출력과 같게 print하고 problem type, file name, search algorithm을 각각 type, file, algorithm 변수에 입력 받는다.

```python
if type < 1 or type > 2 or algorithm < 1 or algorithm > 3:
    print("Error: Input number should be in range.")

if type == 2 and algorithm == 3:
    print("Error: TSP cannot be solved by Gradient Descent.")

if type == 1:
    p = problem.Numeric()
else:
    p = problem.Tsp()
```

```python
p.setVariables(file)

if algorithm == 1:
    a = optimizer.SteepestAscent()
    a.steepestAscent(p)

elif algorithm == 2:
    a = optimizer.FirstChoice()
    a.firstChoice(p)

else:
    a = optimizer.GradientDescent()
    a.gradientDescent(p)
```

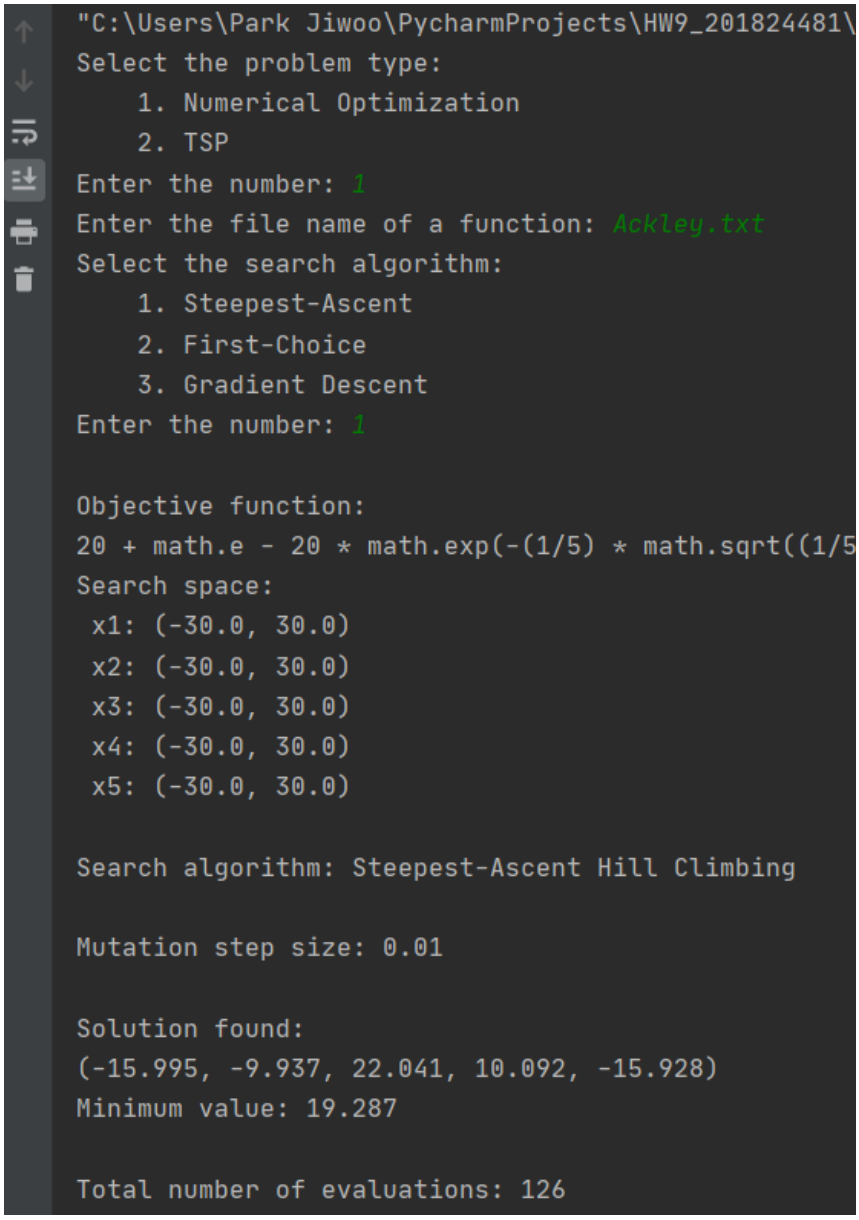
Type이 1보다 작거나 2보다 큰 경우, algorithm이 1보다 작거나 3보다 큰 경우 선택지가 해당되지 않으므로 에러 메시지를 출력한다. 또한 TSP 문제는 Gradient-Descent로 풀 수 없으므로 에러 메시지를 출력한다. 두 에러 모두 발생하지 않으면, p를 Numeric이나 Tsp 클래스 변수로 선언한다. 그후 앞서 입력 받은 파일 이름을 인자로 p의 값을 초기화한다. 선택된 알고리즘에 따라 Steepest-Ascent, First-Choice, Gradient-Descent 방식에 따라 문제를 푼다.

```
41  p.describe()
42  a.displaySetting()
43
44  if type == 1 and (algorithm == 1 or algorithm == 2):
45      a.displayDelta()
46  elif algorithm == 3:
47      a.displayUpdateDx()
48  p.report()
```

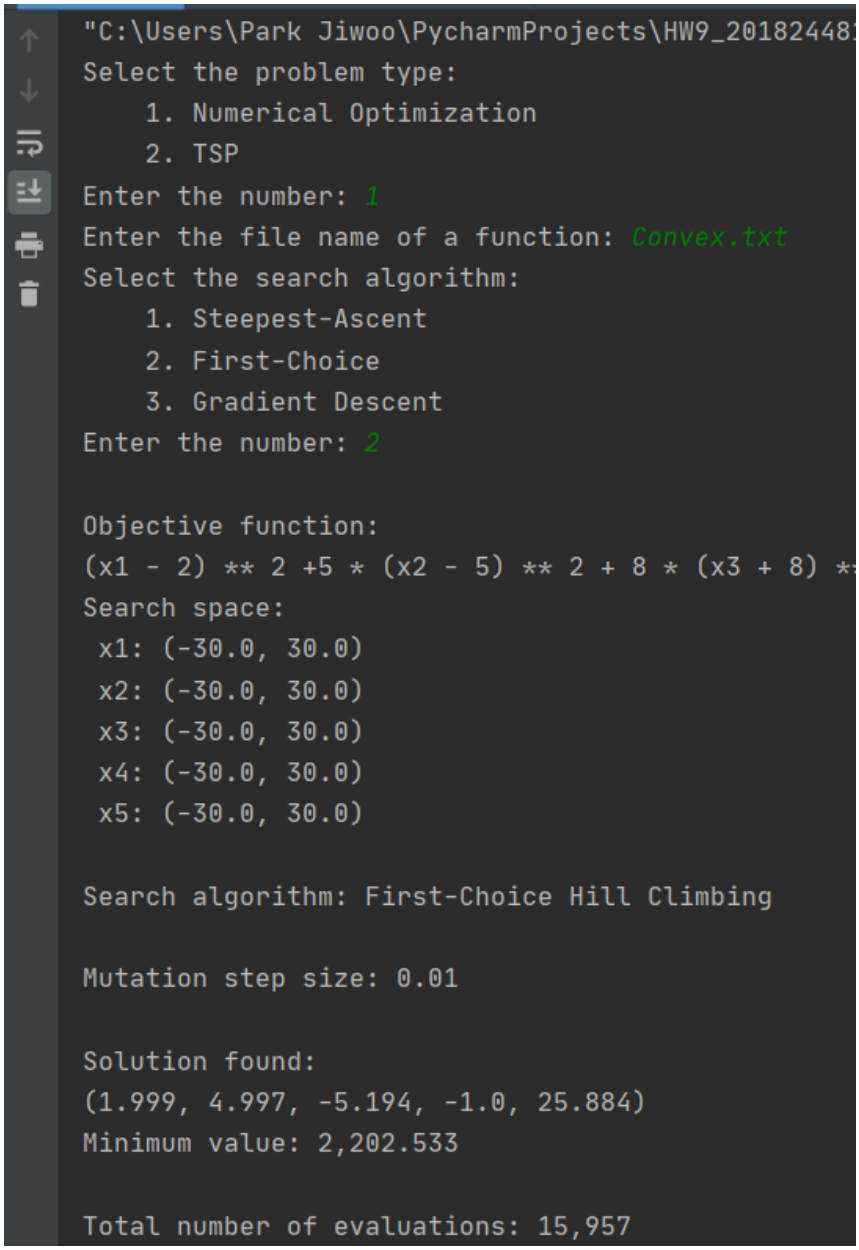
저장된 결과를 출력한다. 만약 Numeric이고 Steepest-Ascent이거나 First-Choice이면 delta값을 출력해주고 Gradient-Descent라면 updateRate와 dx를 출력한다.

5. 출력 결과

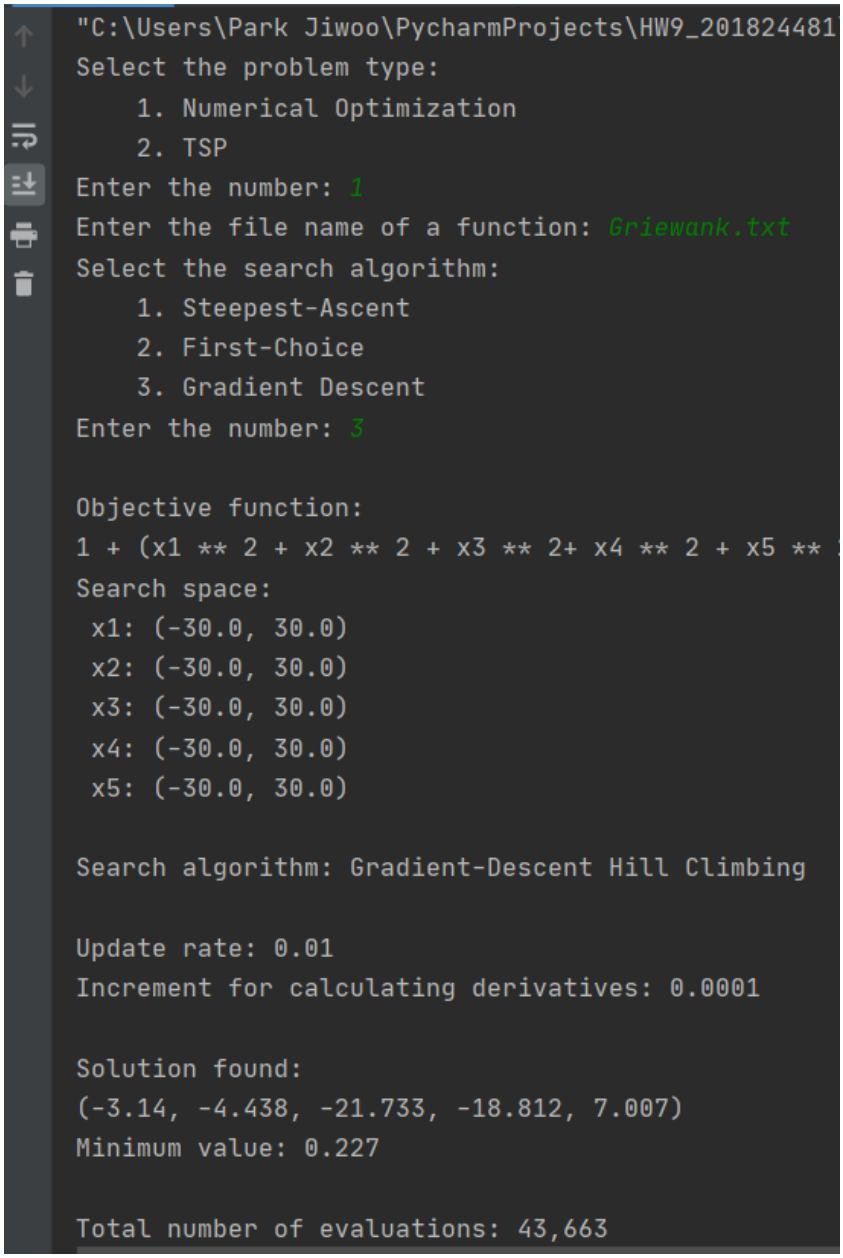
Numeric의 경우 수식이 너무 길어 보고서에 제출하기 번거로워 사진에 모두 담기지 못했다.



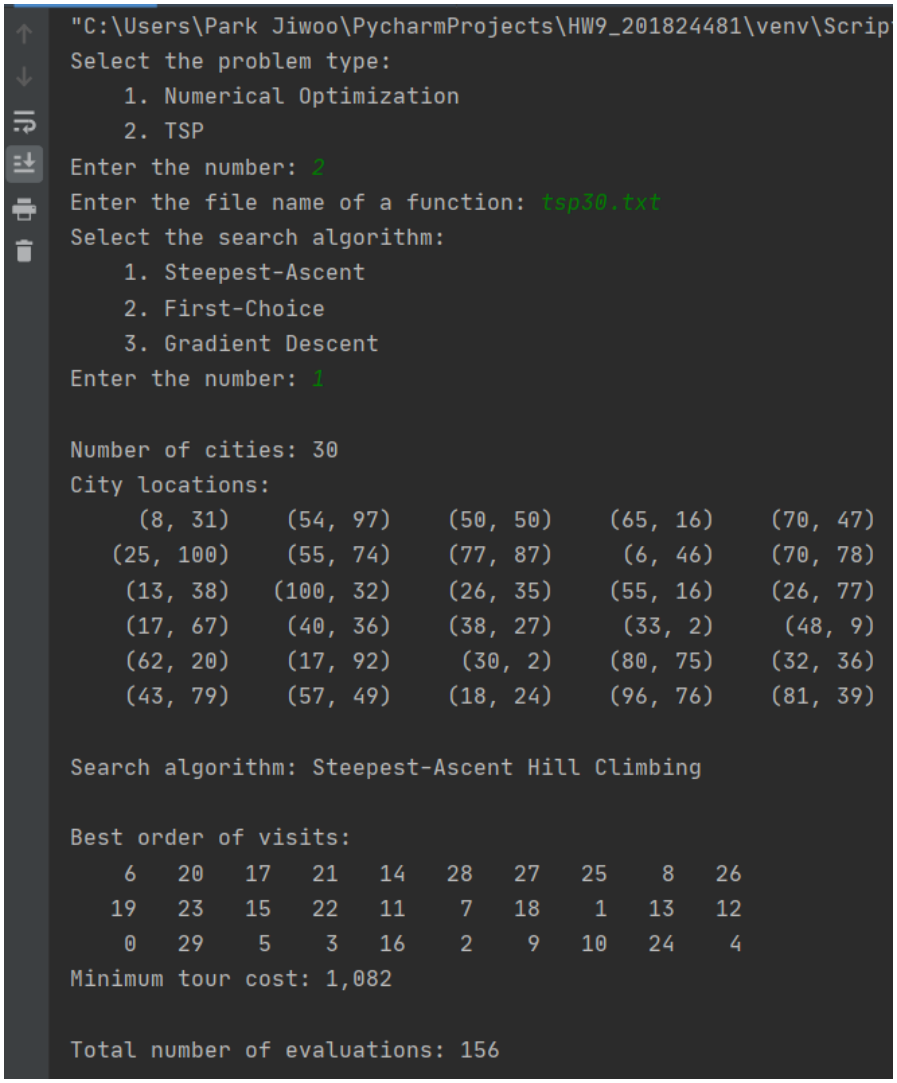
Numeric과 Steepest-Ascent의 결과이다.



Numeric과 First-Choice의 결과이다.



Numeric과 Gradient-Descent의 결과이다.

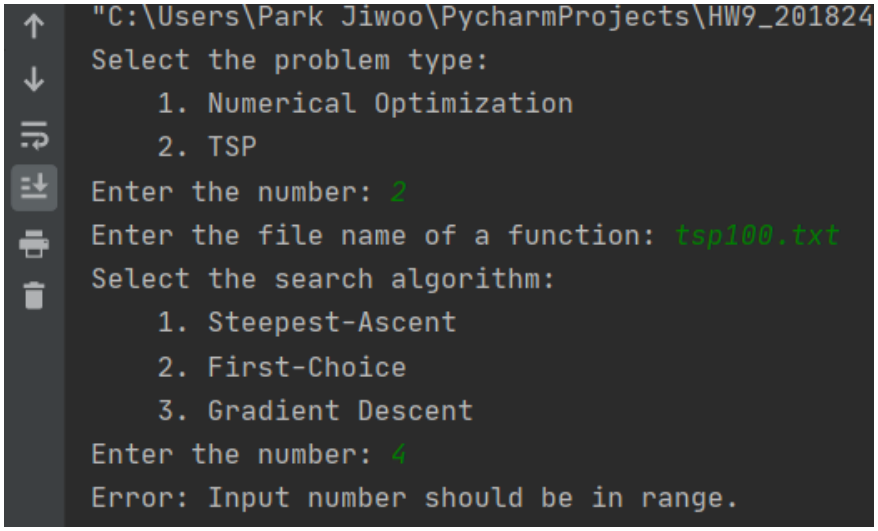


TSP와 Steepest-Ascent의 결과이다.

```
Enter the number: 2
Enter the file name of a function: tsp50.txt
Select the search algorithm:
    1. Steepest-Ascent
    2. First-Choice
    3. Gradient Descent
Enter the number: 2

Number of cities: 50
City locations:
      (1, 7)    (14, 92)    (45, 97)    (17, 60)    (22, 44)
     (4, 38)    (13, 73)    (79, 68)    (76, 95)    (62, 14)
    (25, 75)     (26, 9)    (88, 81)    (56, 65)    (64, 71)
    (92, 20)     (7, 20)     (8, 20)    (61, 39)    (17, 11)
    (10, 40)    (18, 72)    (89, 72)    (58, 25)    (57, 57)
    (66, 70)    (36, 72)    (89, 91)    (18, 90)    (72, 49)
    (82, 38)    (22, 26)    (36, 56)    (23, 44)    (45, 45)
     (7, 27)     (84, 6)    (32, 78)     (0, 29)    (64, 63)
    (45, 24)    (21, 81)    (37, 16)    (86, 57)    (65, 99)
    (25, 53)    (98, 24)    (83, 81)     (50, 5)    (58, 80)

Search algorithm: First-Choice Hill Climbing

Best order of visits:
   13   27   30   40   41   34   28   33    8   10
   37   17   32   19   42   22   35   14   25   15
   36   38    7   21   31    3   45    2   49    0
   11   44   39   18   20   46   23    5    9    6
   43   29   12    1   48    4   26   24   16   47
Minimum tour cost: 1,798

Total number of evaluations: 289
```

TSP와 First-Choice의 결과이다.

```
"C:\Users\Park Jiwoo\PycharmProjects\HW9_2018244
Select the problem type:
    1. Numerical Optimization
    2. TSP
Enter the number: 2
Enter the file name of a function: tsp100.txt
Select the search algorithm:
    1. Steepest-Ascent
    2. First-Choice
    3. Gradient Descent
Enter the number: 3
Error: TSP cannot be solved by Gradient Descent.
```



TSP와 Gradient-Descent를 고른 경우와 선택지에 없는 번호를 골랐을 때 에러 메시지가 출력된다.