|  |
| --- |
| *Pencarian nilai minimum fungsi dengan algoritma Simulated Annealing (SA)* |
| Laporan Tugas Program 1 Kecerdasan Buatan |
| Dosen Pengampu:  Untari Novia Wisesty  Disusun oleh :  Febry Ghaisani (1301154576) |

9/16/2017

**2017**

# Spesifikasi Program

1. Studi Kasus

Membangun program pencarian nilai minimum dari fungsi

dengan batasan dan dengan menerapkan algoritma ***Simulated Annealing*** (SA).

1. Batasan Masalah

* Program dibangun menggunakan Bahasa pemrograman Python.
* Program menggunakan algoritma *Simulated Annealing*.
* Tidak ada inputan user, untuk inisialisasi dengan membangkitkan bilangan acak.
* Nilai minimum yang dihasilkan tidak bersifat *exact* tetapi bersifat *approximately*.

1. Spesifikasi Kebutuhan Program

* PC/Laptop
* OS Minimum Windows 7
* Python 2.7
* JetBrains PyCharm
* Anaconda 2

# Rancangan Metode

Algoritma *Simulated Annealing* (SA) :

1. Evaluasi *initial state*. Jika *state* ini adalah *goal state*, maka kembalikan *state* ini sebagai solusi dan keluar dari program. Jika bukan, lanjutkan dengan *initial state* sebagai *current state*.
2. Inisialisasi *BEST-SO-FAR* dengan *current state*.
3. Inisialisasi T sesuai dengan *annealing schedule*.
4. Ulangi sampai solusi ditemukan atau tidak ada lagi aturan produksi yang bisa diaplikasikan ke *current state* :
5. Pilih sebuah aturan produksi yang belum pernah diaplikasikam ke *current state* dan aplikasikan aturan produksi tersebut untuk menghasilkan *new state*.
6. Evaluasi *new state*. Hitung :

* Jika *new state* adalah *goal*, maka kembalikan *state* ini sebagai solusi dan keluar dari program.
* Jika *new state* bukan *goal* tetapi lebih baik daripada *current state* (), maka set *current state* ke *new state*. Juga set *BEST-SO-FAR* ke *new state*.
* Jika *new state* tidak lebih baik daripada *current state* (), maka set *current state* ke *new state* dengan probabilitas p sebagai berikut :
* Langkah ini biasanya diimplementasikan dengan membangkitkan sebuah bilangan acak dalam interval [0,1]. Jika bilangan tersebut kurang dari p, maka set *current state* ke *new state*. Jika bilangan tersebut lebih besar daripada p, maka jangan mengerjakan apapun.

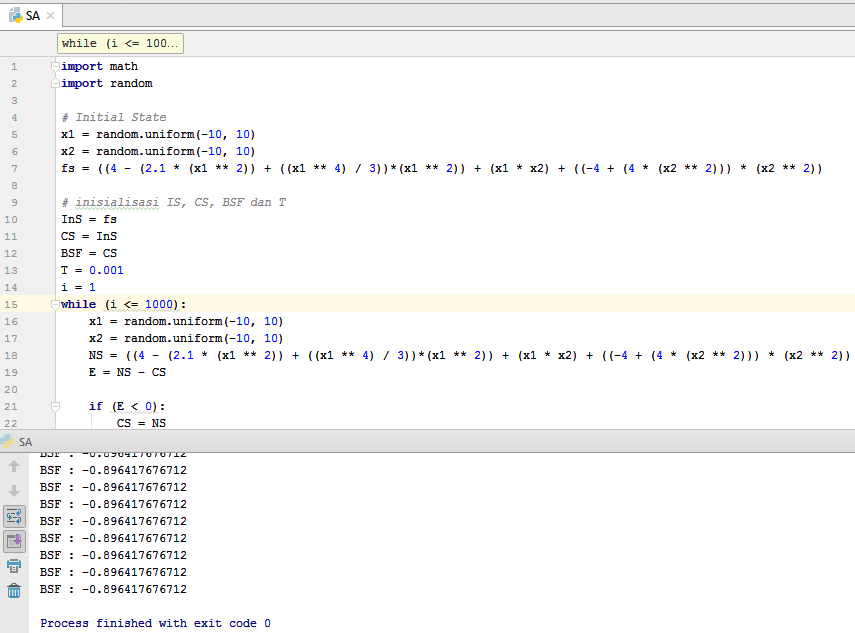
1. Jika diperlukan, revisi nilai T berdasarkan *annealing schedule*.
2. Kembalikan *BEST-SO-FAR* sebagai solusi.

# Pengujian Program

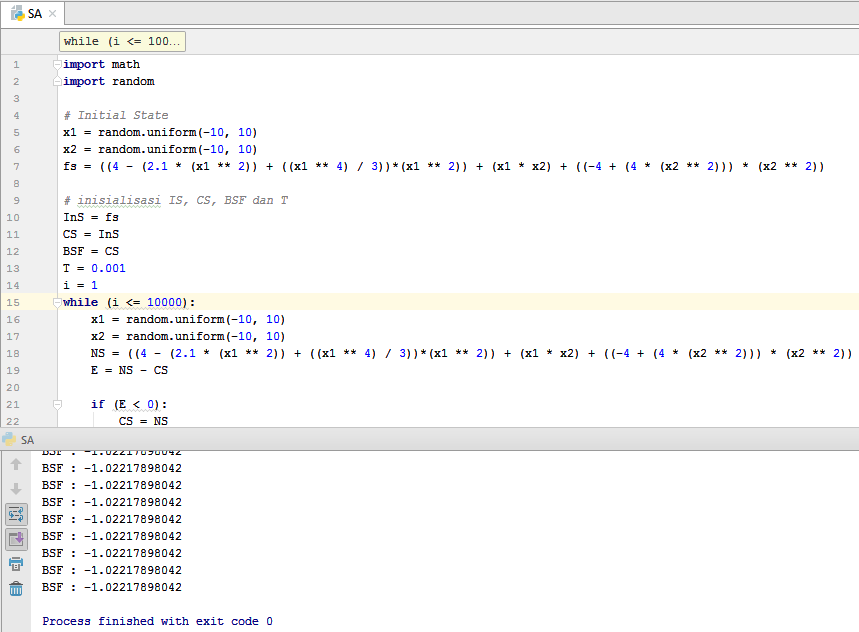
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Jumlah Iterasi** | **Nilai T** | | | | | |
| 0.001 | 0.01 | 0.1 | 1 | 10 | 100 |
| 1.000 | - 0,869417 | - 0,694868 | - 1,019792 | 0,263334 | - 1,008681 | 2,432535 |
| 10.000 | - 1,022178 | - 1,015769 | - 0,803842 | - 0,337107 | - 0,897150 | 4,298456 |
| 100.000 | - 1,029196 | - 1,022557 | - 1,010631 | - 0,926010 | 1,034026 | 70,673537 |

1. Untuk T = 0.001

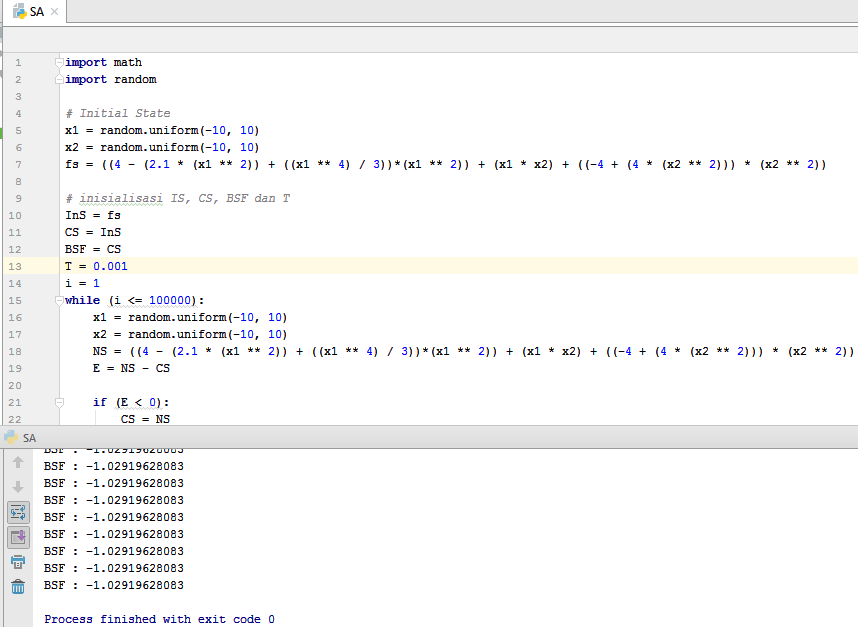
Jumlah Iterasi = 1.000



Jumlah Iterasi = 10.000

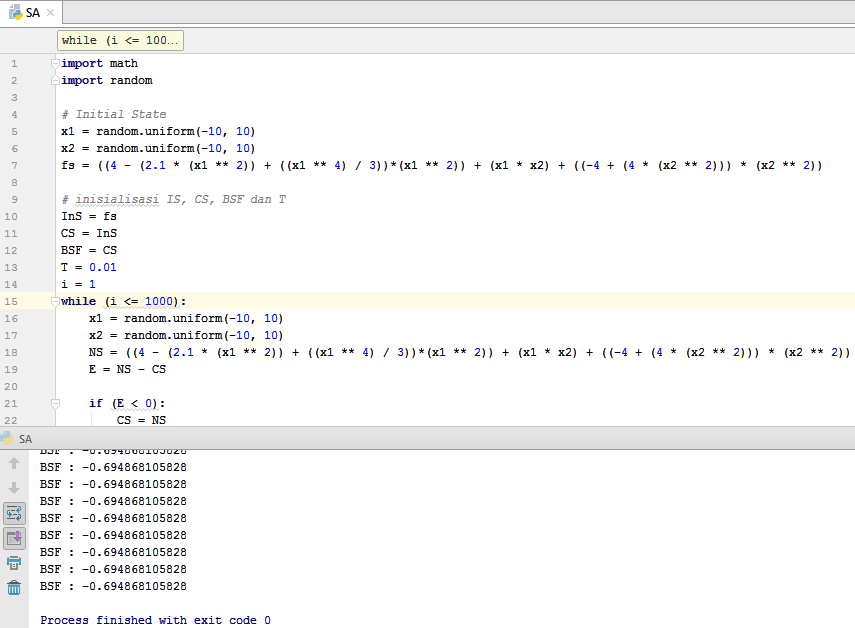


Jumlah Iterasi = 100.000

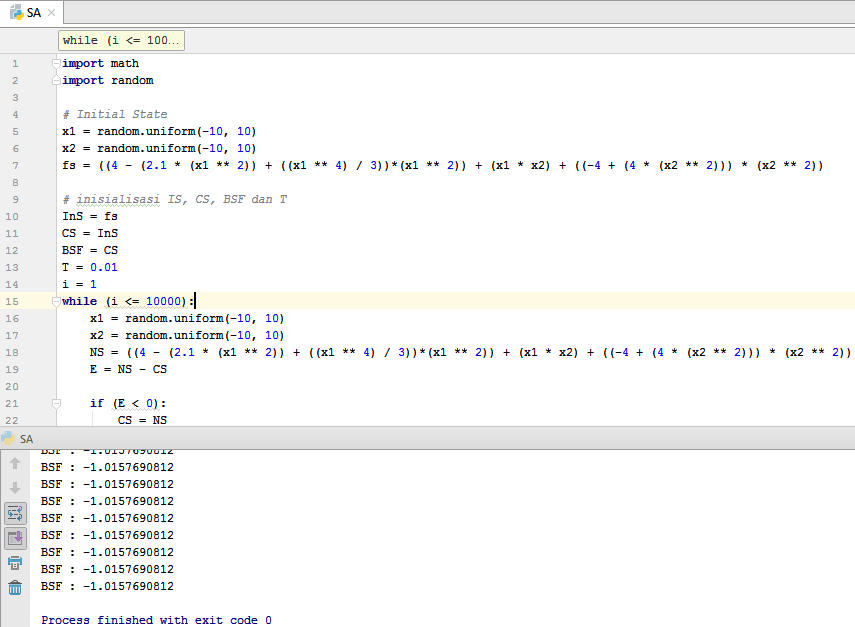


1. Untuk T = 0.01

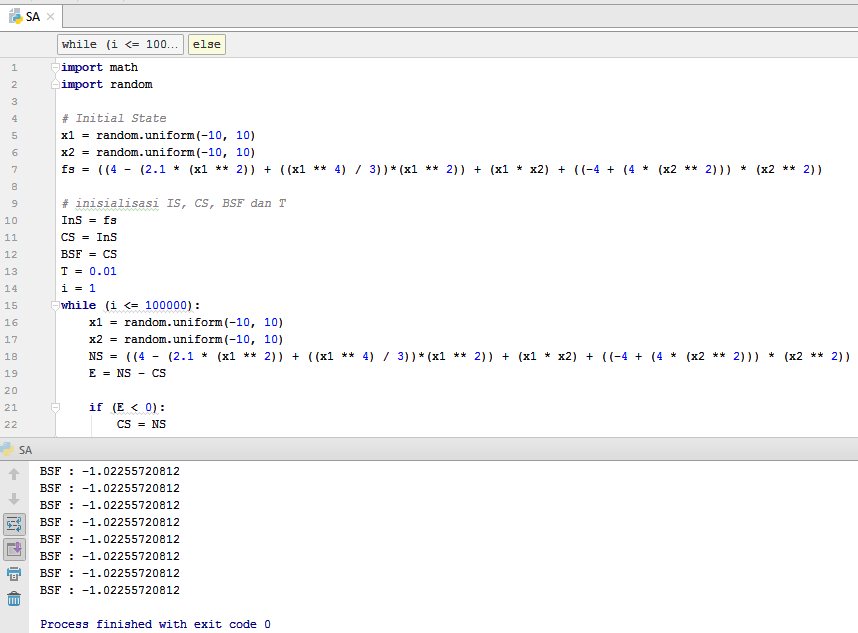
Jumlah iterasi = 1.000



Jumlah iterasi = 10.000

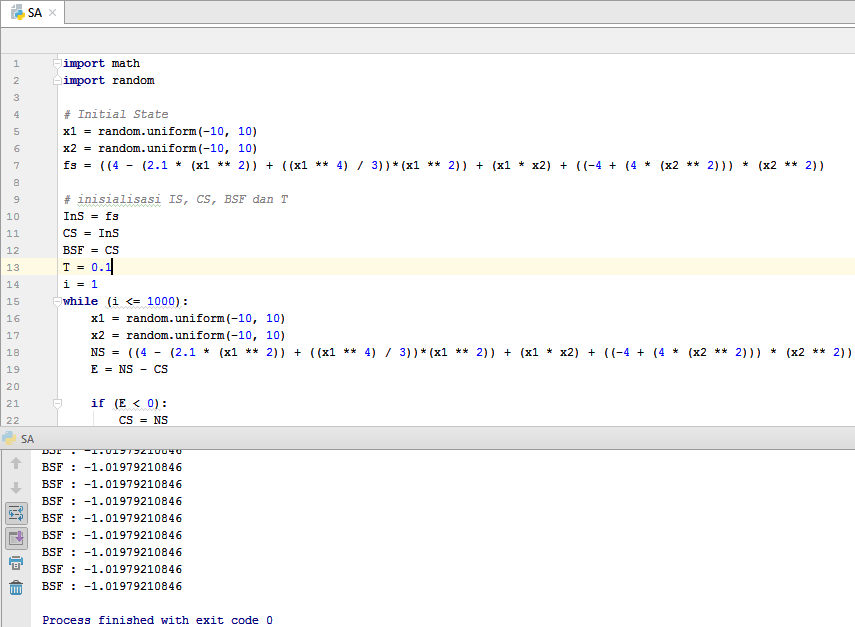


Jumlah Iterasi = 100.000

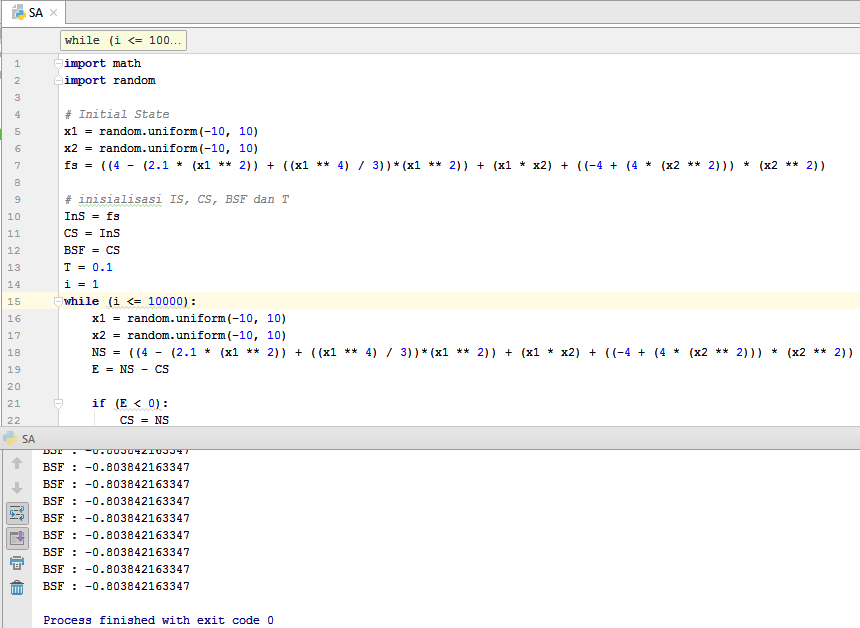


1. Untuk T = 0.1

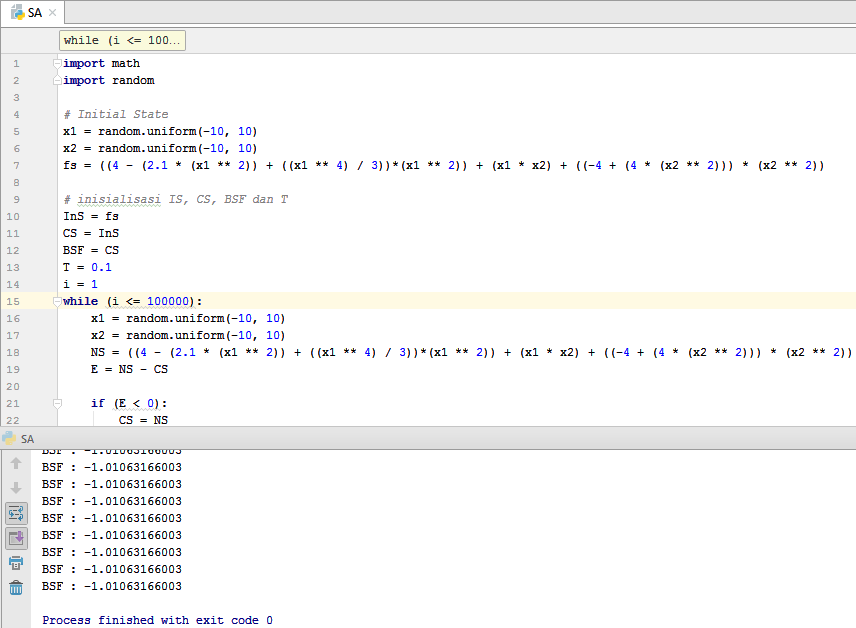
Jumlah Iterasi = 1.000



Jumlah Iterasi = 10.000

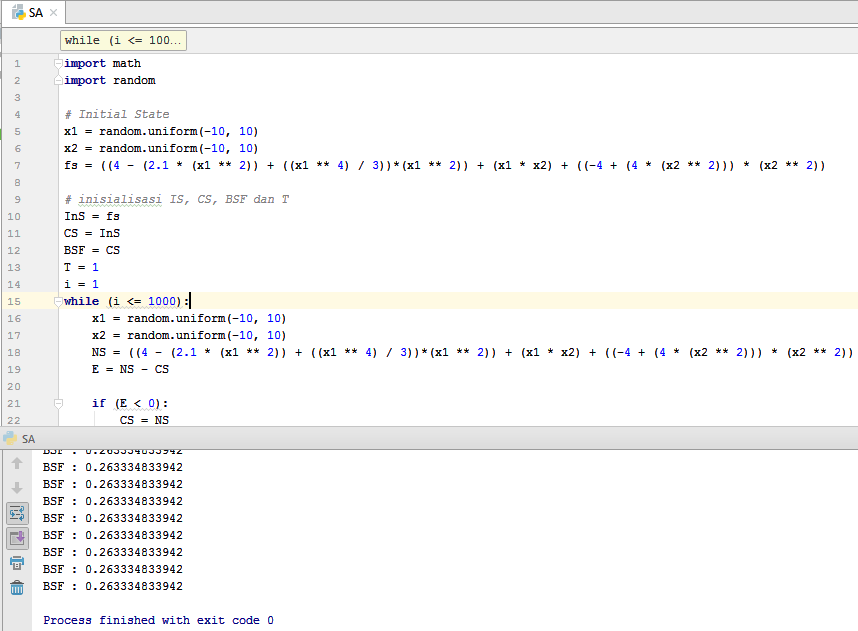


Jumlah Iterasi = 100.000

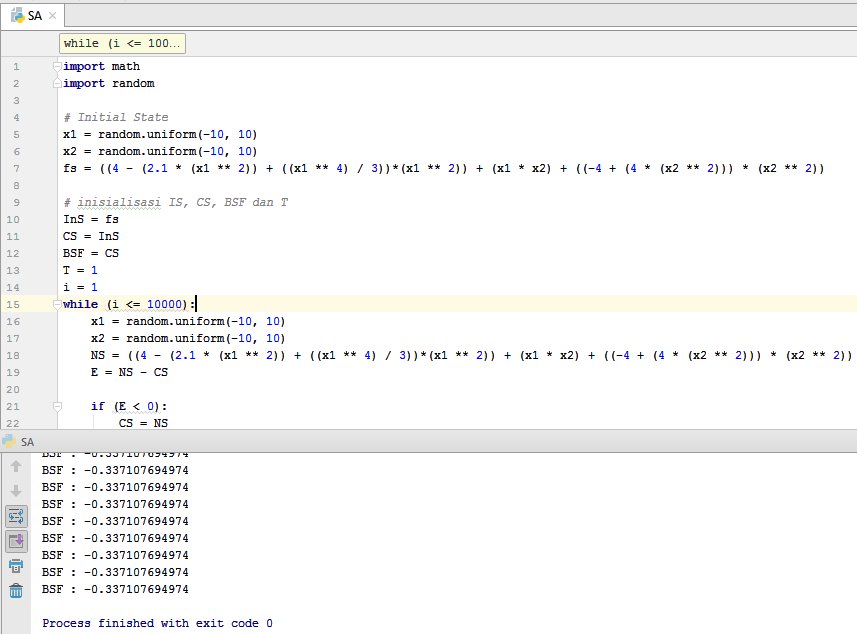


1. Untuk T = 1

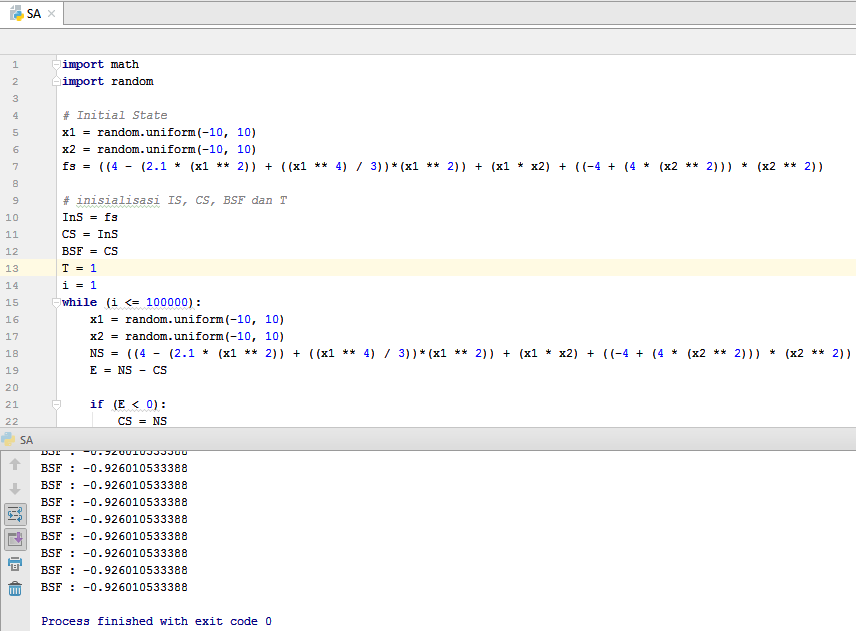
Jumlah Iterasi = 1.000



Jumlah Iterasi = 10.000

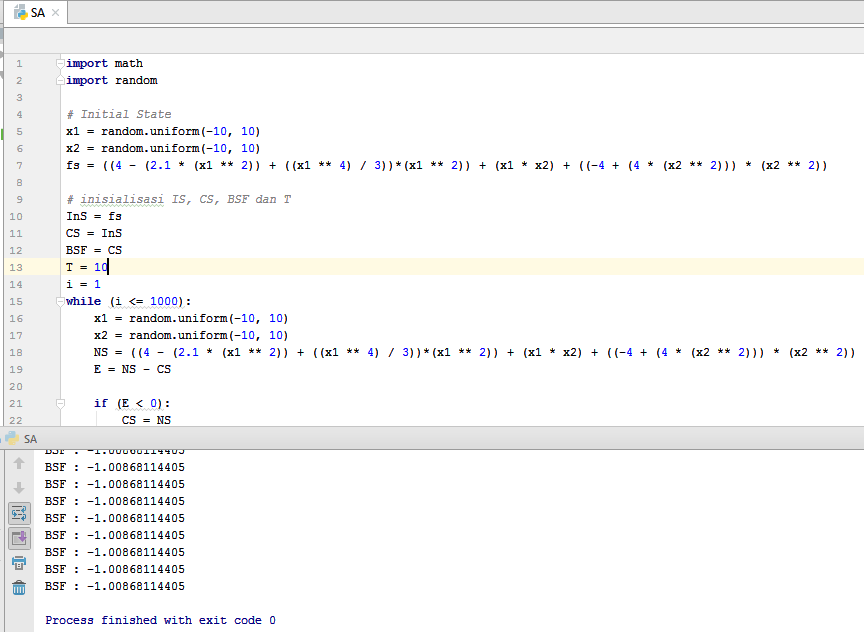


Jumlah Iterasi = 100.000

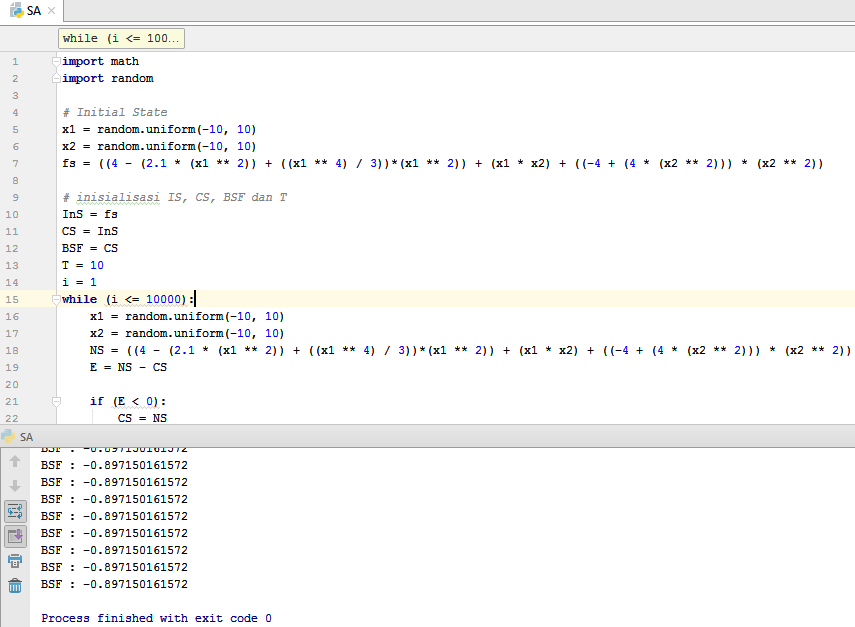


1. Untuk T = 10

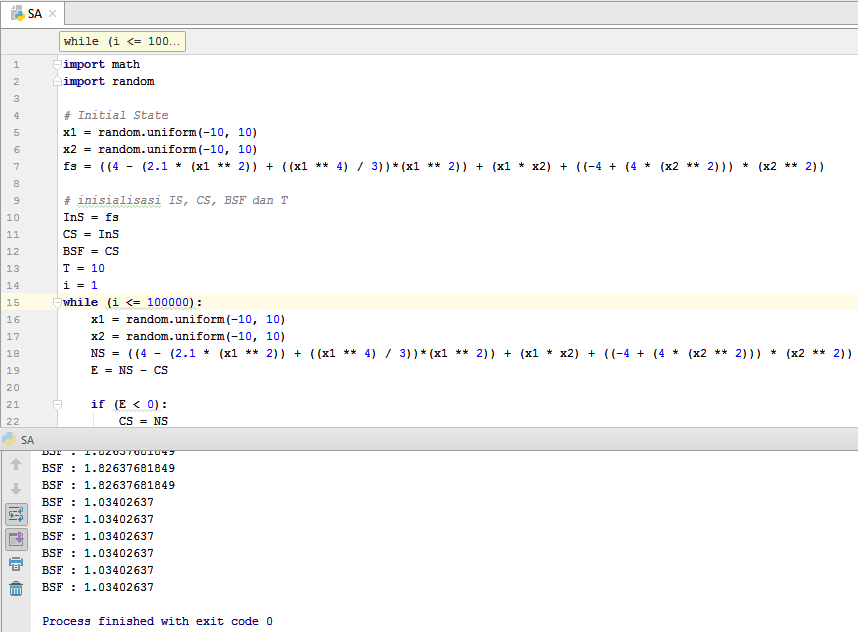
Jumlah Iterasi = 1.000



Jumlah Iterasi = 10.000

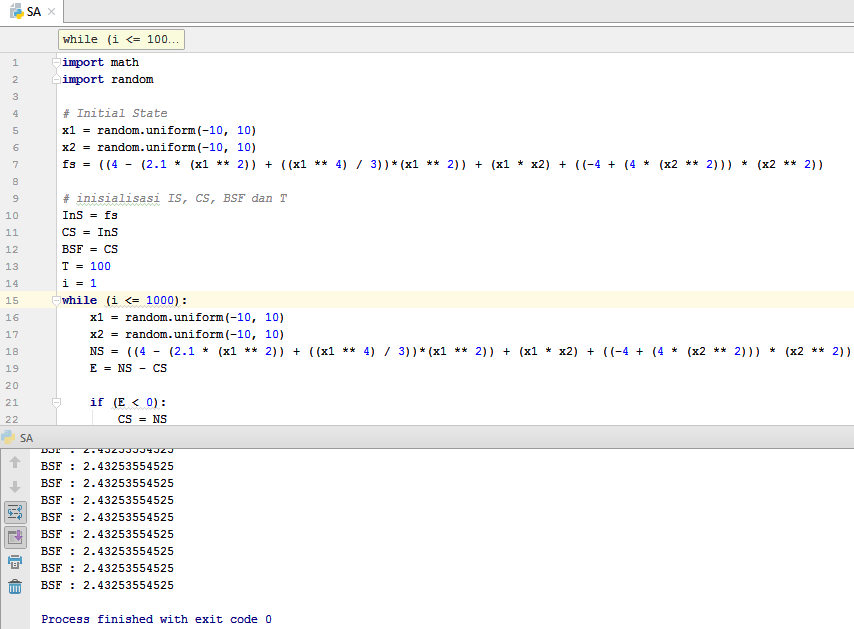


Jumlah Iterasi = 100.000

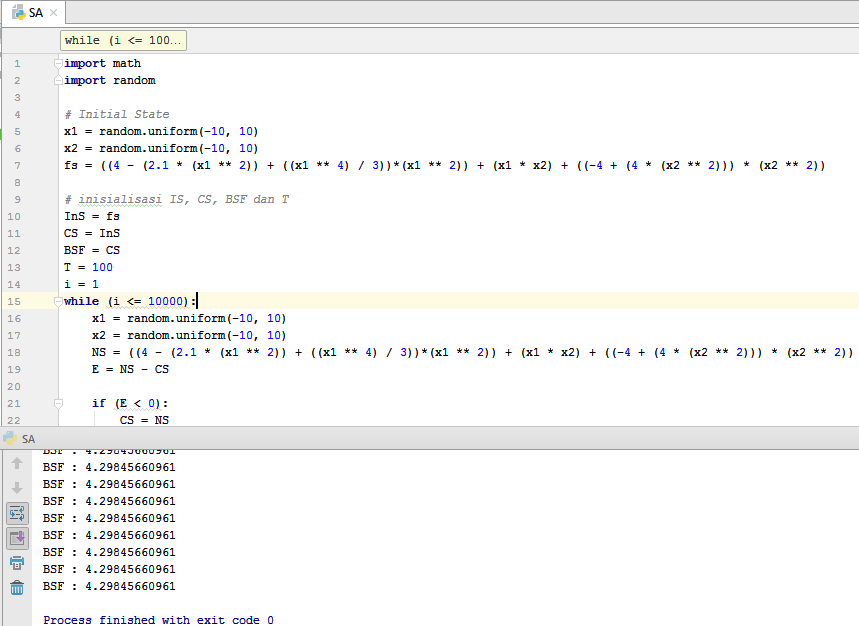


1. Untuk T = 100

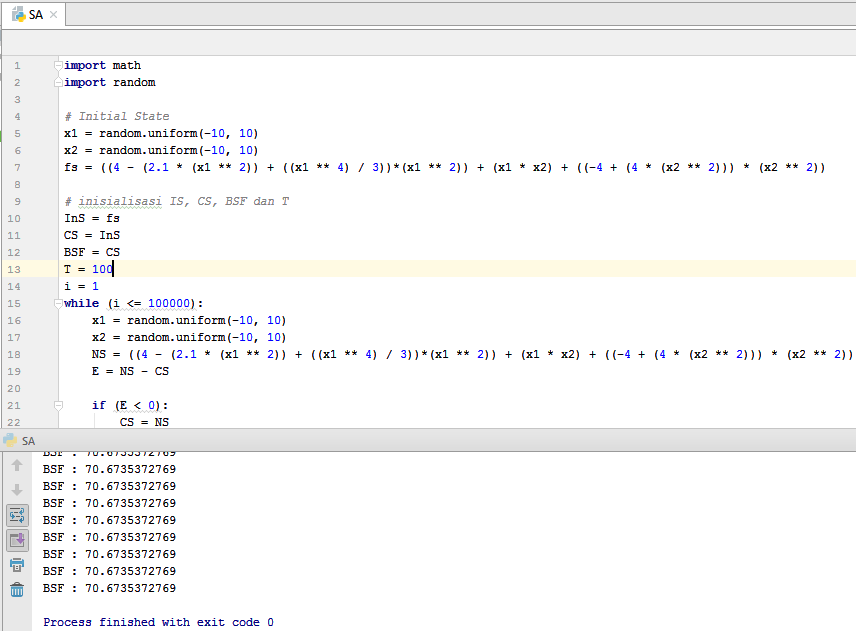
Jumlah Iterasi = 1.000



Jumlah Iterasi = 10.000

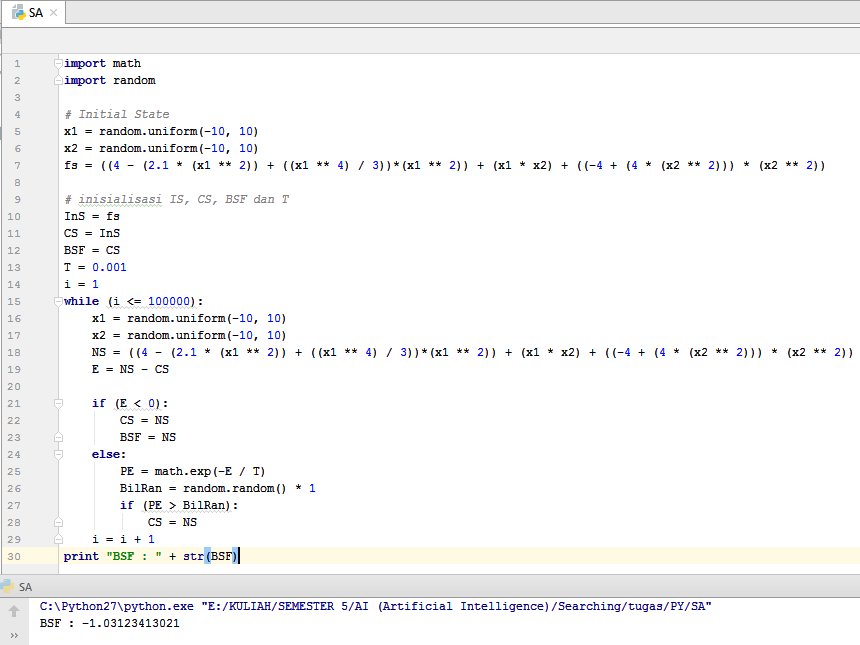


Jumlah Iterasi = 100.000



# Screenshot Program

Screenshot program yang telah dibangun beserta outputnya:



# Referensi

Suyanto. (2014). *Artificial Intelligence Revisi Kedua.* Bandung: Informatika Bandung.

Slide mata kuliah Artificial Intelligence Bab 04 – Heuristic Search.