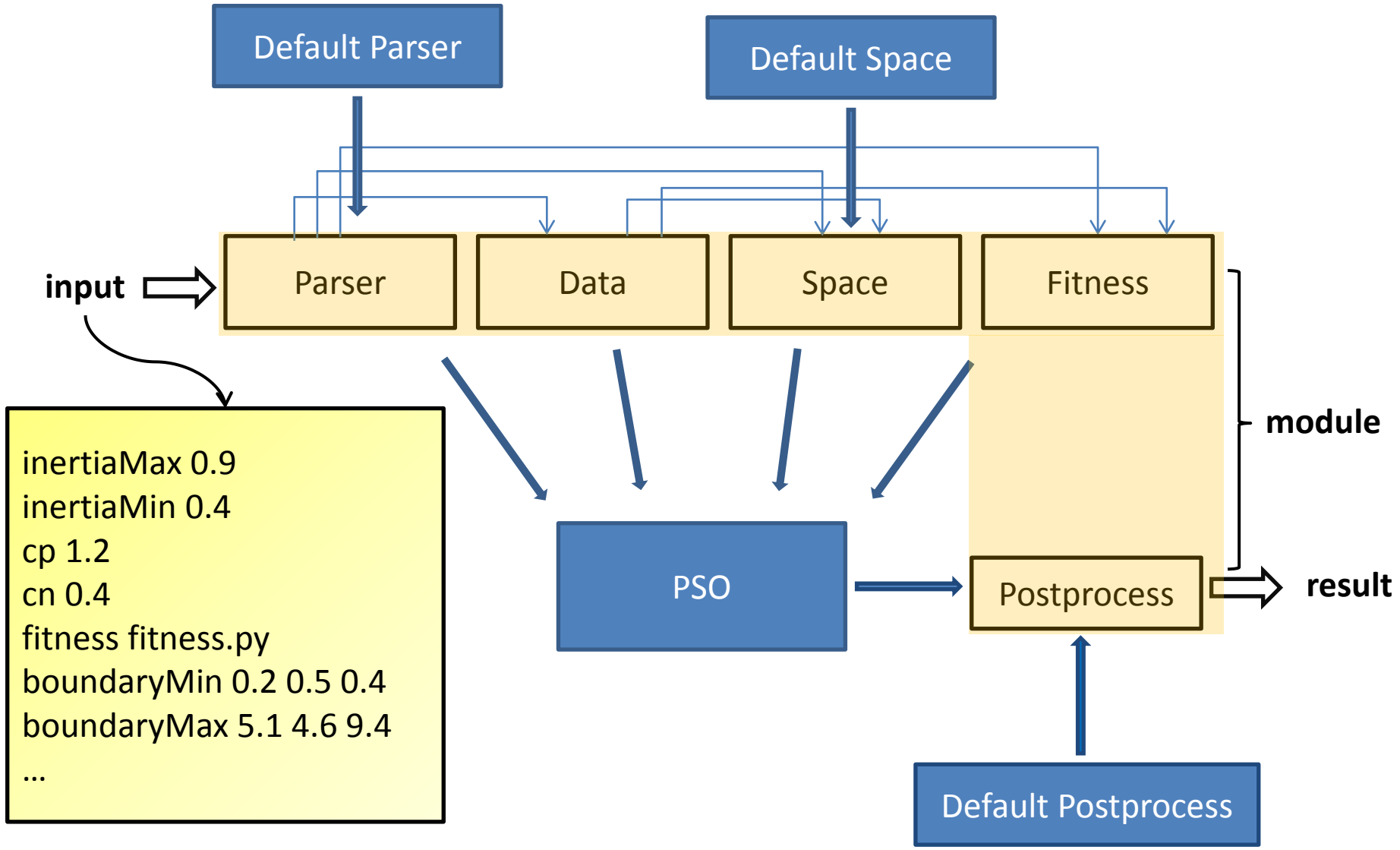

Default Parser
Default Space
input
Parser
Data
Space
Fitness
module
inertiaMax 0.9
inertiaMin 0.4
cp 1.2
cn 0.4
fitness fitness.py
boundaryMin 0.2 0.5 0.4
boundaryMax 5.1 4.6 9.4
…
PSO
Postprocess
result
Default Postprocess

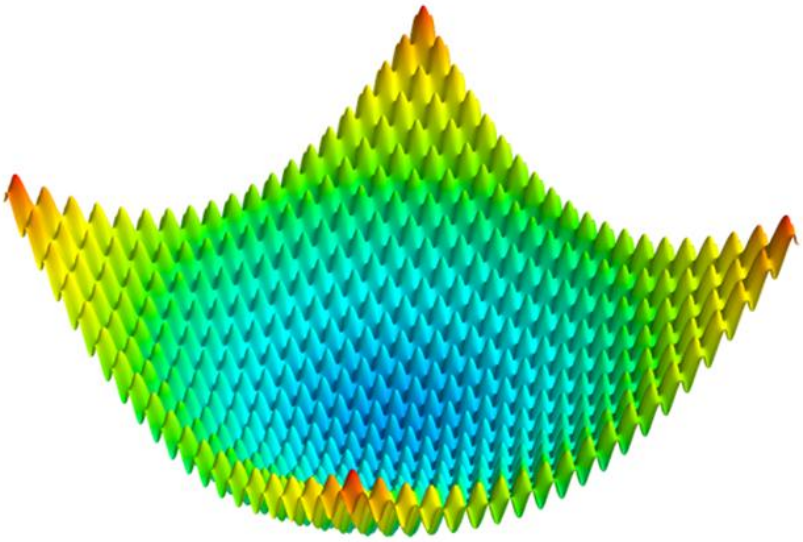
a module skeleton

```
from Default import Parser as R
from Default import Space as S
from Default import Postprocess as PP
# import other packages here

class Parser(R):
        def __init__(self, infile):
                #if needed, parse non-default parametrs
class Data:
        def __init__(self, params):
                #if needed, load files (using your parsed information contained in params object)
class Space(S):
        def __init__(self, params, data):
                #build search space using params and loaded data objects defining the following arrays:
                #self.low = low boundaries
                #self.high = high boundaries
                #self.boundary_type = array having 0 for periodic, and 1 for reflexive boundaries
class Fitness:
        def __init__(self, data, params):
                #if needed, load data here (e.g. target measures,...)
        def evaluate(self, num, x):
                #return fitness value

class Postprocess(PP):
        def __init__(self, data, params):
                #if needed, load relevant data here
        def run(log):
                #launch postprocessing
```

a basic example: Rastringin function

```
from Default import Parser as R
from Default import Space as S
From Default import Postprocess as PP
import numpy as np

class Parser(R):
        def __init__(self,infile):
                pass

class Data:
        def __init__(self,params):
                pass

class Space(S):
        def __init__(self,params,data):
                self.low=np.zeros(len(params.low_input))
                for i in xrange(0,len(params.low_input),1):
                        self.low[i]=params.low_input[i]
                self.high=np.zeros(len(params.low_input))
                for i in xrange(0,len(params.high_input),1):
                        self.high[i]=params.high_input[i]
                self.boundary_type=np.zeros(len(params.low_input))
                for i in xrange(0,len(params.low_input),1):
                        self.boundary_type[i]=params.boundary_type[i]
```

```
class Fitness:
        def __init__(self,data,params):
                pass
        def evaluate(self,num,x):
                return 10*len(x)+np.sum(x**2-10*np.cos(2*np.pi*x))

class Postprocess(PP):
        def __init__(self, data, params):
                pass
        def run(log):
                pass
```



$f(x)=10n+\Sigma_n (x_n^2-10*\cos(2\pi x_n)); f(0)=0$