Programowanie strukturalne w Pythonie

Obliczanie całki z funkcji jednej zmiennej metodą trapezów na siatce nierównomiernej integral.py

<sxh python> """Calculate integral. """ import sys

def main():

```
[fun, a, b] = parse_command_line(sys.argv)
mesh = read_mesh("mesh.dat")
a = mesh[0][0]
b = mesh[-1][0]
integral = integrate(mesh, fun)
```

```
print("Integral of %s in [%g %g] is %g" %(fun, a, b, integral))
```

def parse_command_line(argv):

```
"""Parse command line"""
argc = len(sys.argv)
if len(sys.argv) > 1:
  fun = sys.argv[1]
else:
  print("Using default function f(x) = x^2")
  fun = "x**2"
```

```
if len(sys.argv) > 3:
  a = float(sys.argv[2])
  b = float(sys.argv[3])
else:
  print("Using default domian [0,1]")
  a = 0.0
  b = 1.0
```

return [fun, a, b]

def read_mesh(filename):

```
"""Read from file a list of nodes"""
nodes = list()
with open(filename, 'r') as f:
  for l in f:
    nodes.append(tuple(float(x) for x in l.split()))
return nodes
```

def integrate(mesh, fun):

```
"""Integrate function fun using numerical integration on given mesh"""
mesh_fun = make_mesh_fun(mesh, fun)
```

```
nelem = len(mesh)-1
integral = 0.0
for i in range(nelem):
  integral = integrate_element(i, mesh, mesh_fun)
return integral
```

def make_mesh_fun(mesh, fun):

```
xcoord = [ node[0] for node in mesh ]
discrete_fun = [ eval(fun) for x in xcoord ]
return discrete_fun
```

def integrate_element(i, mesh, mesh_fun):

```
x1 = mesh[i][0]
x2 = mesh[i+1][0]
f1 = mesh_fun[i]
f2 = mesh_fun[i+1]
h = x2 - x1
integral = h * (f1+f2) / 2.0
return integral
```

if name == 'main':

```
main()
```

</sxh>