Anejo 1: Código subprograma 1: Ventana de entrada de datos (ISC_v0.py)

```
# -*- coding: utf-8 -*-
"""
Spyder Editor
Python 3.5.2

Primera part del codi del programa Inlet Spacing Calculator (ISC)
Finestra d'entrada de dades
Versió 0

22/09/2017, Barcelona
Autor: Miquel Sàrrias
"""

import os
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter import PhotoImage
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib#per importar graf a tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg#per importar graf a tk
from matplotlib.figure import Figure#per importar graf a tk
matplotlib.use('TkAgg')#per importar graf a tk
import time



var=[0]*23 #vector variables càlcul
var[0]=2

########## Definició de funcions Tkinter

def geom_method(self):

    frame3s1=ttk.Frame(frame3)
    frame3s1.grid(column=1, row=2, columnspan=4, rowspan=5)

    x=geom.get()

    if x == 'Parameters A and B known':

        def helpAB():
            messagebox.showinfo('Help', 'A and B parameters are obtained from lab tests. \
            They can be related with the efficiency of the inlet.')

        def grafAB():

            A=float(entrya.get())
            B=float(entryb.get())
            cf=float(entrycf.get())

            var[14]=A
            var[15]=B
            var[16]=cf

            Sx=float(entrysx1.get())
            So=float(entryso.get())
            n= float(entryn1.get())

            qq = np.arange(0, 1, 0.01)
            ee = []
            cc = []
```

```
65
66              for i in range(len(qq)):
67                  e=A*np.power(np.power(qq[i],5/8)/np.power(Sx*n/0.376/np.sqrt
    (So),3/8),-B)
68                  if e < 1:
69                      ee.append(e)
70                  else:
71                      ee.append(1)
72
73              for i in range(len(qq)):
74                  cc.append(ee[i]*(1-cf))
75
76              f = Figure(figsize=(2.76, 3.25), dpi=100)
77              a = f.add_subplot(111)
78              a.plot(qq, ee)
79              a.plot(qq, cc, 'r--')
80              a.axis([0, 1, 0, 1.1], fontsize=5)
81              a.set_title('Efficiency', fontsize=10)
82              a.set_xlabel('Flow (m3/s)', fontsize=10)
83              a.text(0.5, 0.9, r'A = %.3f'%(A), fontsize=10)
84              a.text(0.5, 0.8, r'B = %.3f'%(B), fontsize=10)
85              canvas = FigureCanvasTkAgg(f, frameimg3)
86              canvas.show()
87              canvas.get_tk_widget().grid(column=0, row=0)
88
89
90              return
91
92          frame3s1.destroy()
93          frame3s1=ttk.Frame(frame3)
94          frame3s1.grid(column=1, row=2, columnspan=4, rowspan=5)
95
96          labela = ttk.Label(frame3s1, text='A:').grid(column=1, row=1, padx=
    (30,10), pady=50)
97          labelb = ttk.Label(frame3s1, text='B:').grid(column=3, row=1, padx=
    (30,10), pady=50)
98          labelcf = ttk.Label(frame3s1, text='Clogging coefficient:')
99          labelcf.grid(column=1, columnspan=3, row=2, padx=(0,10), pady=0,
    sticky='E')
100
101
102         geoma = StringVar()
103         geomb = StringVar()
104         geomcf = StringVar()
105
106         entrya = ttk.Entry(frame3s1, width=7, textvariable=geoma)
107         entrya.grid(column=2, row=1, pady=50)
108         entryb = ttk.Entry(frame3s1, width=7, textvariable=geomb)
109         entryb.grid(column=4, row=1, pady=50, padx=(0,30))
110         entrycf = ttk.Entry(frame3s1, width=7, textvariable=geomcf)
111         entrycf.grid(column=4, row=2, pady=0, padx=(0,30))
112
113         entrya.focus()
114
115         Button(frame3s1, text='Help', command=helpAB).grid(row=3, column=2,
    columnspan=2, pady=(110,10), padx=(36,0))
116         Button(frame3s1, text='Draw', command=grafAB).grid(row=3, column=3,
    columnspan=2, pady=(110,10), padx=(36,0))
117
118
119 #Gràfica inicial buida
120
121         f = Figure(figsize=(2.76,3.25), dpi=100)
122         a = f.add_subplot(111)
123         t = np.arange(0.0, 3.0, 0.01)
124         s = np.sin(2*np.pi*t)
```

```
125
126         a.plot(t, s, '--w')
127         a.axis([0, 1, 0, 1.1], fontsize=5)
128         a.set_title('Efficiency', fontsize=10)
129         a.set_xlabel('Flow (m3/s)', fontsize=10)
130         a.text(0.5, 0.9, 'A = ', fontsize=10)
131         a.text(0.5, 0.8, 'B = ', fontsize=10)
132
133         canvas = FigureCanvasTkAgg(f, frameimg3)
134         canvas.show()
135         canvas.get_tk_widget().grid(column=0, row=0)
136
137
138     if x == 'Geometry known':
139
140         def helpgk():
141             messagebox.showinfo('Help', 'Posar explicacio de parametres i
    unitats')
142
143         def grafgk():
144
145             Ag=float(entryAg.get())
146             Ah=float(entryAh.get())
147             nt=float(entrynt.get())
148             nl=float(entrynl.get())
149             nd=float(entrynd.get())
150             L=float(entryL.get())
151             W=float(entryW.get())
152             cf=float(entrycf.get())
153
154             Sx=float(entrysx1.get())
155             So=float(entryso.get())
156             n=float(entryn1.get())
157
158             A=1.988*np.power(Ag,0.403)/np.power(Ah/Ag,0.190)/np.power(nt
    +1,0.088)/\
159             np.power(nl+1,0.012)/np.power(nd+1,0.082)
160             B=1.346*np.power(L,0.179)/np.power(W,0.394)
161
162             var[14]=A
163             var[15]=B
164             var[16]=cf
165
166             qq = np.arange(0, 1, 0.01)
167             ee = []
168             cc =[]
169
170             for i in range(len(qq)):
171                 e=A*np.power(np.power(qq[i],5/8)/np.power(Sx*n/0.376/np.sqrt
    (So),3/8),-B)
172                 if e < 1:
173                     ee.append(e)
174                 else:
175                     ee.append(1)
176
177             for i in range(len(qq)):
178                 cc.append(ee[i]*(1-cf))
179
180             f = Figure(figsize=(2.76, 3.25), dpi=100)
181             a = f.add_subplot(111)
182             a.plot(qq, ee)
183             a.plot(qq, cc, 'r--')
184             a.axis([0, 1, 0, 1.1], fontsize=5)
185             a.set_title('Efficiency', fontsize=10)
186             a.set_xlabel('Flow (m3/s)', fontsize=10)
187             a.text(0.5, 0.9, r'A = %.3f'%(A), fontsize=10)
```

```
188                 a.text(0.5, 0.8, r'B = %.3f'%(B), fontsize=10)
189                 canvas = FigureCanvasTkAgg(f, frameimg3)
190                 canvas.show()
191                 canvas.get_tk_widget().grid(column=0, row=0)
192 
193                 return
194 
195             frame3s1.destroy()
196             frame3s1=ttk.Frame(frame3)
197             frame3s1.grid(column=1, row=2, columnspan=4, rowspan=5, sticky='N')
198 
199             geomAg = StringVar()
200             geomAh = StringVar()
201             geomnt = StringVar()
202             geomnl = StringVar()
203             geomnd = StringVar()
204             geomL = StringVar()
205             geomW = StringVar()
206             geomcf = StringVar()
207 
208             entryAg = ttk.Entry(frame3s1, width=7, textvariable=geomAg)
209             entryAg.grid(column=2, row=1, sticky='N', pady=(20,0))
210             entryAh = ttk.Entry(frame3s1, width=7, textvariable=geomAh)
211             entryAh.grid(column=2, row=2)
212             entrynt = ttk.Entry(frame3s1, width=7, textvariable=geomnt)
213             entrynt.grid(column=2, row=3)
214             entrynl = ttk.Entry(frame3s1, width=7, textvariable=geomnl)
215             entrynl.grid(column=2, row=4)
216             entrynd = ttk.Entry(frame3s1, width=7, textvariable=geomnd)
217             entrynd.grid(column=2, row=5)
218             entryL = ttk.Entry(frame3s1, width=7, textvariable=geomL)
219             entryL.grid(column=2, row=6)
220             entryW = ttk.Entry(frame3s1, width=7, textvariable=geomW)
221             entryW.grid(column=2, row=7)
222             entrycf = ttk.Entry(frame3s1, width=7, textvariable=geomcf)
223             entrycf.grid(column=2, row=8, pady=(10,10))
224 
225             labelAg = ttk.Label(frame3s1, text='Total closed area:').grid
    (column=1, row=1, padx=(0,20), pady=(20,5), sticky=('E'))
226             labelAh = ttk.Label(frame3s1, text='Total gap area:').grid(column=1,
    row=2, padx=(0,20), pady=5, sticky='E')
227             labelnt = ttk.Label(frame3s1, text='Num. of tranversal bars:').grid
    (column=1, row=3, padx=(0,20), pady=5, sticky='E')
228             labelnl = ttk.Label(frame3s1, text='Num. of longitudinal bars:').grid
    (column=1, row=4, padx=(0,20), pady=5, sticky='E')
229             labelnd = ttk.Label(frame3s1, text='Num. of diagonal bars:').grid
    (column=1, row=5, padx=(0,20), pady=5, sticky='E')
230             labelL = ttk.Label(frame3s1, text='Inlet lenght:').grid(column=1,
    row=6, padx=(0,20), pady=5, sticky='E')
231             labelW = ttk.Label(frame3s1, text='Inlet width:').grid(column=1,
    row=7, padx=(0,20), pady=5, sticky='E')
232             labelcc = ttk.Label(frame3s1, text='Clogging coefficient:').grid
    (column=1, row=8, padx=(0,20), pady=(10,10), sticky='E')
233 
234             Button(frame3s1, text='Help', command=helpgk).grid(row=9, column=1,
    columnspan=2, pady=(10,10))
235             Button(frame3s1, text='Draw', command=grafgk).grid(row=9, column=2,
    columnspan=2, pady=(10,10))
236 
237     #Gràfica inicial buida
238 
239             f = Figure(figsize=(2.76,3.25), dpi=100)
240             a = f.add_subplot(111)
241             t = np.arange(0.0, 3.0, 0.01)
242             s = np.sin(2*np.pi*t)
243 
```

```
244             a.plot(t, s, '--w')
245             a.axis([0, 1, 0, 1.1], fontsize=5)
246             a.set_title('Efficiency', fontsize=10)
247             a.set_xlabel('Flow (m3/s)', fontsize=10)
248             a.text(0.5, 0.9, 'A = ', fontsize=10)
249             a.text(0.5, 0.8, 'B = ', fontsize=10)
250 
251             canvas = FigureCanvasTkAgg(f, frameimg3)
252             canvas.show()
253             canvas.get_tk_widget().grid(column=0, row=0)
254 
255 
256         if x == '-':
257 
258             frame3s1.destroy()
259             frame3s1=ttk.Frame(frame3)
260             frame3s1.grid(column=1, row=2, columnspan=4, rowspan=5, sticky='N')
261 
262         return
263 
264 
265     def streetcs(self):
266 
267         x=street.get()
268 
269         if x == 'Gutter' or x == 'V-section':
270 
271             labelsx2.configure(state='disabled')
272             labeln2.configure(state='disabled')
273             labelw2.configure(state='disabled')
274             labelk.configure(state='disabled')
275             entrysx2.configure(state='disabled')
276             entryn2.configure(state='disabled')
277             entryw2.configure(state='disabled')
278             entryk.configure(state='disabled')
279 
280         else:
281 
282             labelsx2.configure(state='normal')
283             labeln2.configure(state='normal')
284             labelw2.configure(state='normal')
285             labelk.configure(state='normal')
286             entrysx2.configure(state='normal')
287             entryn2.configure(state='normal')
288             entryw2.configure(state='normal')
289             entryk.configure(state='normal')
290 
291         if x == 'Gutter':
292 
293             photo = PhotoImage(file='IMG_gutter.gif')
294             labelimg = Label(frameimg, image=photo)
295             labelimg.image = photo
296             labelimg.grid(column=0, row=0)
297 
298             var[0]=1
299 
300         if x == 'Gutter with sidewalk':
301 
302             photo = PhotoImage(file='IMG_guttersidewalk.gif')
303             labelimg = Label(frameimg, image=photo)
304             labelimg.image = photo
305             labelimg.grid(column=0, row=0)
306 
307             var[0]=2
308 
309         if x == 'V-section':
```

```
310
311             photo = PhotoImage(file='IMG_vsection.gif')
312             labelimg = Label(frameimg, image=photo)
313             labelimg.image = photo
314             labelimg.grid(column=0, row=0)
315
316             var[0]=3
317
318         if x == 'V-section with sidewalk':
319
320             photo = PhotoImage(file='IMG_vsectionsidewalk.gif')
321             labelimg = Label(frameimg, image=photo)
322             labelimg.image = photo
323             labelimg.grid(column=0, row=0)
324
325             var[0]=4
326
327     def buttonlimc():
328
329         x=limrec.get()
330
331         if x == '1':
332
333             entryymax.delete(0, END)
334             entryvmax.delete(0, END)
335             entryvymax.delete(0, END)
336             entryvy2max.delete(0, END)
337
338             entryymax.insert(0,'0.06')
339             entryvmax.insert(0,'1.80')
340             entryvymax.insert(0,'0.50')
341             entryvy2max.insert(0,'1.00')
342
343             labelymax.configure(state='disable')
344             labelvmax.configure(state='disable')
345             labelvymax.configure(state='disable')
346             labelvy2max.configure(state='disable')
347             entryymax.configure(state='disable')
348             entryvmax.configure(state='disable')
349             entryvymax.configure(state='disable')
350             entryvy2max.configure(state='disable')
351
352         else:
353
354             labelymax.configure(state='normal')
355             labelvmax.configure(state='normal')
356             labelvymax.configure(state='normal')
357             labelvy2max.configure(state='normal')
358             entryymax.configure(state='normal')
359             entryvmax.configure(state='normal')
360             entryvymax.configure(state='normal')
361             entryvy2max.configure(state='normal')
362
363         return
364
365     def drawlim():
366         ymax=float(entryymax.get())
367         vmax=float(entryvmax.get())
368         vymax=float(entryvymax.get())
369         vy2max=float(entryvy2max.get())
370
371         var[10]=ymax
372         var[11]=vmax
373         var[12]=vymax
374         var[13]=vy2max
375
```

```
376
377         xx = np.arange(0, 1, 0.01)
378         vy = []
379         vy2 =[]
380
381         xx2=np.arange(0,ymax,0.01)
382         zz = []
383
384         for i in range(len(xx)):
385             vy.append(vymax/xx[i])
386             vy2.append(vy2max/np.power(xx[i],2))
387
388         for i in range(len(xx2)):
389             zz.append(min(vmax,vy[i],vy2[i]))
390
391
392         f = Figure(figsize=(2.76, 3.25), dpi=100)
393         a = f.add_subplot(111)
394         a.axhline(y=vmax, xmin=0, xmax=1, linewidth=1, color = 'k')
395         a.axvline(x=ymax, ymin=0, ymax=1, linewidth=1, color = 'k')
396         a.plot(xx, vy,  'k')
397         a.plot(xx, vy2, 'k')
398         a.fill_between(xx2,zz, color='c')
399         a.axis([0, 1, 0, 5], fontsize=5)
400         a.set_title('Flow velocity (m/s)', fontsize=10)
401         a.set_xlabel('Flow depth (m)', fontsize=10)
402         red_patch = mpatches.Patch(color='c', label='Accepted area')
403         a.legend(handles=[red_patch], fontsize=9)
404
405         canvas = FigureCanvasTkAgg(f, frameimg2)
406         canvas.show()
407         canvas.get_tk_widget().grid(column=0, row=0)
408
409         return
410
411     def rain_method(self):
412
413         frame4s1=ttk.Frame(frame4)
414         frame4s1.grid(column=1, row=2, columnspan=4, rowspan=5)
415
416         x=rain.get()
417
418         if x == 'T=10 Barcelona':
419
420             def helpidfb():
421                 k=1
422
423     #         frame4s1.destroy()
424     #         frame4s1=ttk.Frame(frame4)
425     #         frame4s1.grid(column=1, row=2, columnspan=4, rowspan=5, sticky='N')
426
427             label = ttk.Label(frame4s1, text=' ')
428             label.grid(column=0, row=0, padx=120, pady=92, columnspan=2)
429
430             Button(frame4s1, text='Help', command=helpidfb).grid(row=2, column=0,
    pady=(10,0), padx=(60,0))
431
432             t=[0, 1/12, 2/12, 3/12, 4/12, 5/12, 6/12, 7/12, 8/12, 9/12, 10/12,
    11/12, 12/12, 13/12, \
433             14/12, 15/12, 16/12, 17/12, 18/12, 19/12, 20/12, 21/12, 22/12, 23/12]
434             rf=
    [4.1,4.8,5.6,6.7,8.1,10.0,12.7,16.6,22.7,32.9,51.7,92.9,135,67,40,27,19,14.5,11.2,9,7.36,6.1
435
436             var[17]=1
437             var[18]=0
438             var[19]=0
```

```
439             var[20]=0
440             var[21]=0
441             var[22]=0
442 
443             f = Figure(figsize=(2.76, 3.25), dpi=100)
444             a = f.add_subplot(111)
445             a.bar(t, rf, width = 1/12)
446             a.axis([0, 2, 0, 140], fontsize=5)
447             a.set_title('Intensity (mm/h)', fontsize=10)
448             a.set_xlabel('Time (h)', fontsize=10)
449             a.grid(True)
450 
451             canvas = FigureCanvasTkAgg(f, frameimg4)
452             canvas.show()
453             canvas.get_tk_widget().grid(column=0, row=0)
454 
455 
456         if x == 'IDF general equation':
457 
458             def helpidf():
459                 k=1
460             def grafidf():
461                 a=float(entryidfa.get())
462                 b=float(entryidfb.get())
463                 c=float(entryidfc.get())
464                 d=float(entryidfd.get())
465                 D=float(entryidfD.get())
466                 At=5
467 
468                 rf1=[]
469                 n=int(D*60/At)
470 
471                 for i in range(0,n):
472                         I=a/np.power(np.power(At*(i+1),c)+b,d)
473                         rf1.append(I*(i+1)-sum(rf1))
474 
475                 t=[]
476                 rf=np.zeros(n)
477 
478                 for i in range(0,n):
479                     t.append(At/60*i)
480                     if i == 0:
481                         rf[n//2]=rf1[0]
482                     if i%2 == 1:
483                         rf[n//2-i//2-1]=rf1[i]
484                     if i%2 == 0:
485                         rf[n//2+i//2]=rf1[i]
486 
487                 var[17]=2
488                 var[18]=a
489                 var[19]=b
490                 var[20]=c
491                 var[21]=d
492                 var[22]=D
493 
494                 f = Figure(figsize=(2.76, 3.25), dpi=100)
495                 a = f.add_subplot(111)
496                 a.bar(t, rf, width = At/60)
497 #                a.axis([0, 2, 0, 140], fontsize=5)
498                 a.set_title('Intensity (mm/h)', fontsize=10)
499                 a.set_xlabel('Time (h)', fontsize=10)
500                 a.grid(True)
501 
502                 canvas = FigureCanvasTkAgg(f, frameimg4)
503                 canvas.show()
504                 canvas.get_tk_widget().grid(column=0, row=0)
```

```
505
506
507
508             frame4s1.destroy()
509             frame4s1=ttk.Frame(frame4)
510             frame4s1.grid(column=1, row=2, columnspan=4, rowspan=5, sticky='N')
511
512             photo = PhotoImage(file='formula.gif')
513             labelimg = Label(frame4s1, image=photo)
514             labelimg.image = photo
515             labelimg.grid(column=0, row=0, columnspan=4, pady=(0,0))
516
517             label = ttk.Label(frame4s1, text='Equation parameters:').grid
    (column=0, row=1, columnspan=4, pady=(10,5), sticky=('w'))
518             label = ttk.Label(frame4s1, text='Event duration:').grid(column=0,
    row=4, columnspan=4, pady=(20,5), sticky=('w'))
519
520             labela = ttk.Label(frame4s1, text='a:').grid(column=0, row=2, padx=
    (30,0), pady=7)
521             labelb = ttk.Label(frame4s1, text='b:').grid(column=0, row=3, padx=
    (30,0), pady=7)
522             labelc = ttk.Label(frame4s1, text='c:').grid(column=2, row=2, padx=
    (35,0), pady=7)
523             labeld = ttk.Label(frame4s1, text='d:').grid(column=2, row=3, padx=
    (35,0), pady=7)
524             labelD = ttk.Label(frame4s1, text='D:').grid(column=0, row=5, padx=
    (30,0), pady=7)
525 #            labelAt = ttk.Label(frame4s1, text='At:').grid(column=0, row=6, padx=
    (30,0), pady=7)
526
527             idfa = StringVar()
528             idfb = StringVar()
529             idfc = StringVar()
530             idfd = StringVar()
531             idfD = StringVar()
532 #            idfAt = StringVar()
533
534             entryidfa = ttk.Entry(frame4s1, width=7, textvariable=idfa)
535             entryidfa.grid(column=1, row=2, pady=0)
536             entryidfb = ttk.Entry(frame4s1, width=7, textvariable=idfb)
537             entryidfb.grid(column=1, row=3, pady=0)
538             entryidfc = ttk.Entry(frame4s1, width=7, textvariable=idfc)
539             entryidfc.grid(column=3, row=2, pady=0)
540             entryidfd = ttk.Entry(frame4s1, width=7, textvariable=idfd)
541             entryidfd.grid(column=3, row=3, pady=0)
542             entryidfD = ttk.Entry(frame4s1, width=7, textvariable=idfD)
543             entryidfD.grid(column=1, row=5, pady=0)
544 #            entryidfAt = ttk.Entry(frame4s1, width=7, textvariable=idfAt)
545 #            entryidfAt.grid(column=1, row=6, pady=0)
546
547             Button(frame4s1, text='Help', command=helpidf).grid(row=7, column=0,
    columnspan=2, pady=(10,0), padx=(0,0))
548             Button(frame4s1, text='Draw', command=grafidf).grid(row=7, column=2,
    columnspan=2, pady=(10,0), padx=(0,0))
549
550
551
552
553         if x == 'Syntetic IDF':
554             print('syntetic')
555
556         return
557
558
559
560
```

```
561
562  def helplim():
563      messagebox.showinfo('Help', 'limits in meters, m/s and m/s2')
564
565  def togglef2():
566      n.select(f2)
567      print(so,sx1,n1)
568      return
569  def togglef3():
570      n.select(f3)
571      return
572  def togglef4():
573      n.select(f4)
574      return
575
576  def compute():
577
578      txt1=open('ComputingParameters.txt', 'w')
579
580  #    Button4.configure(state='disable')
581
582      if var[0] == 1 or var[0] == 3:
583          so=float(entryso.get())
584          sx1=float(entrysx1.get())
585          n1=float(entryn1.get())
586          w1=float(entryw1.get())
587
588          var[1]=so
589          var[2]=sx1
590          var[3]=n1
591          var[4]=w1
592          var[5]=0
593          var[6]=0
594          var[7]=0
595          var[8]=0
596          var[9]=sl
597
598      if var[0] == 2 or var[0] == 4:
599          so=float(entryso.get())
600          sx1=float(entrysx1.get())
601          sx2=float(entrysx2.get())
602          n1=float(entryn1.get())
603          n2=float(entryn2.get())
604          w1=float(entryw1.get())
605          w2=float(entryw2.get())
606          k=float(entryk.get())
607          sl=float(entrysl.get())
608
609          var[1]=so
610          var[2]=sx1
611          var[3]=n1
612          var[4]=w1
613          var[5]=sx2
614          var[6]=n2
615          var[7]=w2
616          var[8]=k
617          var[9]=sl
618
619  #    txt1=open('ComputingParameters.txt', 'w')
620
621      txt1.write('STREET GEOMETRY\nclass= '+str(var[0])+'\nso   = '+str(var[1])
     +'\nsx1  = '+str(var[2])+\
622      '\nn1   = '+str(var[3])+'\nw1   = '+str(var[4])+'\nsx2  = '+str(var[5])
     +'\nn2   = '+str(var[6])+\
623      '\nw2   = '+str(var[7])+'\nk    = '+str(var[8])+'\nsl   = '+str(var[9])
     +'\n\n')
```

```
624     txt1.write('HAZARD CRITERIA\ny    = '+str(var[10])+'\nv    = '+str(var
    [11])+'\nvy   = '+str(var[12])+\
625     '\nvy2  = '+str(var[13])+'\n\n')
626     txt1.write('INLET GEOMETRY\nA    = '+str(var[14])+'\nB    = '+str(var[15])
    +'\ncc   = '+str(var[16])+'\n\n')
627     txt1.write('RAINFALL\nIM   = '+str(var[17])+'\na    = '+str(var[18])
    +'\nb    = '+str(var[19])+\
628     '\nc    = '+str(var[20])+'\nd    = '+str(var[21])+'\nD    = '+str(var[22])
    +'\n\n')
629
630     txt1.write(time.strftime("Process started at %d %b %Y %H:%M:%S\n",
    time.localtime()))
631
632     txt1.close()
633
634     #os.system('ISC_v0_calc.py')
635     import ISC_v0_calc
636
637
638
639 ##########################################################################################
640 ##########                                D E F I N I C I O N S   W I D G E T
    S                                          ##########
641 ##########################################################################################
642
643 root = Tk()
644 root.title("Inlet spacing calculator")
645 root.geometry("635x455+300+300") #num auri (1+sqrt5)/2=1.6180
646
647 ###############################################################################
    Creació frame principal
648
649 mainframe = ttk.Frame(root, padding="1 2 12 12")
650 mainframe.grid(column=0, row=0, sticky=('N, W, E, S'))
651 mainframe.columnconfigure(0, weight=1)
652 mainframe.rowconfigure(0, weight=1)
653
654 ###############################################################################
    Creació pestanyes en el frame principal
655
656 n=ttk.Notebook(mainframe)
657 f1=ttk.Frame(n)
658 f2=ttk.Frame(n)
659 f3=ttk.Frame(n)
660 f4=ttk.Frame(n)
661 n.add(f1, text="Street Geometry")
662 n.add(f2, text="Hazard Criteria")
663 n.add(f3, text="Inlet Geometry")
664 n.add(f4, text="Rainfall")
665
666 ###############################################################################
    Widgets primera pestanya (STREET GEOMETRY)
667
668 def helpstreet():
669     so=float(entryso.get())
670     sx1=float(entrysx1.get())
671     sx2=float(entrysx2.get())
672     n1=float(entryn1.get())
673     n2=float(entryn2.get())
674     w1=float(entryw1.get())
675     w2=float(entryw2.get())
676     k=float(entryk.get())
677
678     return(so,sx1,sx2,n1,n2,w1,w2,k)
679
680
```

```
681
682   frame1 = ttk.Frame(f1, padding="10 10 12 12")
683   frame1.grid(column=0, row=0)
684   frame1['width'] = 580
685   frame1['height'] = 430
686
687   Button1=ttk.Button(frame1, text="Next", command=togglef2).grid(column=6,
      row=9, pady=(20,0), sticky="E")
688
689   frameimg = ttk.Frame(frame1)
690   frameimg.grid(column=5, row=0, columnspan=2, rowspan=9, padx=(20,0))
691   frameimg['borderwidth'] = 2
692   frameimg['relief'] = 'groove'
693   frameimg['width'] = 300
694   frameimg['height'] = 350
695
696   streetvar = StringVar()
697   street = ttk.Combobox(frame1, textvariable=streetvar, state="readonly")
698   street['values'] = ('Gutter', 'Gutter with sidewalk', 'V-section', 'V-section
      with sidewalk')
699   street.grid(column=1, row=1, columnspan=4)
700   street.current(1)
701   street.bind("<<ComboboxSelected>>", streetcs)
702
703   labeles = ttk.Label(frame1, text='Choose a street cross section:').grid
      (column=1, row=0, columnspan=4, sticky='S')
704   labelso = ttk.Label(frame1, text='So:').grid(column=2, row=2, padx=(40,2),
      sticky='E')
705   labelsx1 = ttk.Label(frame1, text='Sx1:').grid(column=1, row=4, padx=(10,2),
      sticky='E')
706   labelsx2 = ttk.Label(frame1, text='Sx2:')
707   labelsx2.grid(column=3, row=4, padx=(40,2), sticky='E')
708   labelbor = ttk.Label(frame1, text='Road:').grid(column=1, row=3, columnspan=2)
709   labelcal = ttk.Label(frame1, text='Sidewalk:').grid(column=3, row=3,
      columnspan=2)
710   labeln1 = ttk.Label(frame1, text='n1:').grid(column=1, row=5, padx=(10,2),
      sticky='E')
711   labeln2 = ttk.Label(frame1, text='n2:')
712   labeln2.grid(column=3, row=5, padx=(40,2), sticky='E')
713   labelw1 = ttk.Label(frame1, text='w1:').grid(column=1, row=6, padx=(10,5),
      sticky='E')
714   labelw2 = ttk.Label(frame1, text='w2:')
715   labelw2.grid(column=3, row=6, padx=(40,2), sticky='E')
716   labelk = ttk.Label(frame1, text='k:')
717   labelk.grid(column=3, row=7, padx=(40,2), sticky='E')
718   labelsl = ttk.Label(frame1, text='Street length:')
719   labelsl.grid(column=1, row=8, columnspan=2, padx=(40,2), sticky='E')
720
721   so = StringVar()
722   sx1 = StringVar()
723   sx2 = StringVar()
724   n1 = StringVar()
725   n2 = StringVar()
726   w1 = StringVar()
727   w2 = StringVar()
728   k = StringVar()
729   sl = StringVar()
730
731   entryso = ttk.Entry(frame1, width=7, textvariable=so)
732   entryso.grid(column=3, row=2)
733   entrysx1 = ttk.Entry(frame1, width=7, textvariable=sx1)
734   entrysx1.grid(column=2, row=4)
735   entrysx2 = ttk.Entry(frame1, width=7, textvariable=sx2)
736   entrysx2.grid(column=4, row=4, padx=(0,20))
737   entryn1 = ttk.Entry(frame1, width=7, textvariable=n1)
738   entryn1.grid(column=2, row=5)
```

```
739   entryn2 = ttk.Entry(frame1, width=7, textvariable=n2)
740   entryn2.grid(column=4, row=5, padx=(0,20))
741   entryw1 = ttk.Entry(frame1, width=7, textvariable=w1)
742   entryw1.grid(column=2, row=6)
743   entryw2 = ttk.Entry(frame1, width=7, textvariable=w2)
744   entryw2.grid(column=4, row=6, padx=(0,20))
745   entryk = ttk.Entry(frame1, width=7, textvariable=k)
746   entryk.grid(column=4, row=7, padx=(0,20))
747   entrysl = ttk.Entry(frame1, width=7, textvariable=sl)
748   entrysl.grid(column=3, row=8, columnspan=2, padx=(5,0), sticky='W')
749
750   photo = PhotoImage(file='IMG_guttersidewalk.gif')
751   labelimg = Label(frameimg, image=photo)
752   labelimg.image = photo
753   labelimg.grid(column=0, row=0)
754
755   Button(frame1, text='Help', command=helpstreet).grid(row=9, column=2,
      columnspan=2, padx=(60,0), pady=(20,0))
756
757
758
759   ################################################################################
      Widgets segona pestanya (HAZARD CRITERIA)
760
761   frame2 = ttk.Frame(f2, padding="10 10 12 12")
762   frame2.grid(column=0, row=0)
763   frame2['width'] = 580
764   frame2['height'] = 430
765
766   Button2=ttk.Button(frame2, text="Next", command=togglef3).grid(column=6,
      row=7, pady=(20,0), sticky="E")
767
768   frameimg2 = ttk.Frame(frame2, padding="10 10 10 10")
769   frameimg2.grid(column=5, row=0, columnspan=2, rowspan=7, padx=(20,0))
770   frameimg2['borderwidth'] = 2
771   frameimg2['relief'] = 'groove'
772   frameimg2['width'] = 300
773   frameimg2['height'] = 350
774
775   limrec = StringVar()
776   buttonlim = ttk.Checkbutton(frame2, text='Use recommended values',
      command=buttonlimc, variable=limrec, onvalue=1, offvalue=0)
777   buttonlim.grid(column=1, row=1, columnspan=2, padx=(44,45), pady=30)
778
779   labelymax = ttk.Label(frame2, text='y max:')
780   labelymax.grid(column=1, row=2, sticky='E')
781   labelvmax = ttk.Label(frame2, text='v max:')
782   labelvmax.grid(column=1, row=3, sticky='E')
783   labelvymax = ttk.Label(frame2, text='v·y max:')
784   labelvymax.grid(column=1, row=4, sticky='E')
785   labelvy2max = ttk.Label(frame2, text='v·y\u00B2 max:')
786   labelvy2max.grid(column=1, row=5, sticky='E')
787
788   ymax = StringVar()
789   vmax = StringVar()
790   vymax = StringVar()
791   vy2max = StringVar()
792
793   entryymax = ttk.Entry(frame2, width=7, textvariable=ymax)
794   entryymax.grid(column=2, row=2)
795   entryvmax = ttk.Entry(frame2, width=7, textvariable=vmax)
796   entryvmax.grid(column=2, row=3)
797   entryvymax = ttk.Entry(frame2, width=7, textvariable=vymax)
798   entryvymax.grid(column=2, row=4)
799   entryvy2max = ttk.Entry(frame2, width=7, textvariable=vy2max)
800   entryvy2max.grid(column=2, row=5)
```

```
801
802   Button(frame2, text='Help', command=helplim).grid(row=6, column=1,
      columnspan=2, pady=(60,0))
803   Button(frame2, text='Draw', command=drawlim).grid(row=6, column=2,
      columnspan=2, pady=(60,0))
804
805   #Gràfica inicial buida
806
807   f = Figure(figsize=(2.76,3.25), dpi=100)
808   a = f.add_subplot(111)
809   t = np.arange(0.0, 3.0, 0.01)
810   s = np.sin(2*np.pi*t)
811
812   a.plot(t, s, '--w')
813   a.axis([0, 1.2, 0, 5], fontsize=5)
814   a.set_title('Flow velocity (m/s)', fontsize=10)
815   a.set_xlabel('Flow depth (m)', fontsize=10)
816
817   canvas = FigureCanvasTkAgg(f, frameimg2)
818   canvas.show()
819   canvas.get_tk_widget().grid(column=0, row=0)
820
821
822   ##############################################################################
      Widgets tercera pestanya (INLET GEOMETRY)
823
824   frame3 = ttk.Frame(f3, padding="10 10 12 12")
825   frame3.grid(column=0, row=0)
826   frame3['width'] = 580
827   frame3['height'] = 430
828
829   Button3=ttk.Button(frame3, text="Next", command=togglef4).grid(column=6,
      row=7, pady=(20,0), sticky="E")
830
831   frameimg3 = ttk.Frame(frame3, padding="10 10 10 10")
832   frameimg3.grid(column=5, row=0, columnspan=2, rowspan=7, padx=(20,0))
833   frameimg3['borderwidth'] = 2
834   frameimg3['relief'] = 'groove'
835   frameimg3['width'] = 300
836   frameimg3['height'] = 350
837
838   labeles3 = ttk.Label(frame3, text='Choose geometry input method:')
839   labeles3.grid(column=1, row=0, columnspan=4, pady=(15,0), padx=(33,32),
      sticky='N')
840   geomvar = StringVar()
841   geom = ttk.Combobox(frame3, textvariable=geomvar, state="readonly")
842   geom['values'] = ('-','Parameters A and B known', 'Geometry known')
843   geom.grid(column=1, row=0, columnspan=4, pady=(50,0), sticky='N')
844   geom.current(0)
845   geom.bind("<<ComboboxSelected>>", geom_method)
846
847
848   ##############################################################################
      Widgets quarta pestanya (RAINFALL)
849
850   frame4 = ttk.Frame(f4, padding="10 10 12 12")
851   frame4.grid(column=0, row=0)
852   frame4['width'] = 580
853   frame4['height'] = 430
854
855   Button4=ttk.Button(frame4, text="Compute", command=compute).grid(column=6,
      row=7, pady=(20,0), sticky="E")
856   #Button4h=ttk.Button(frame4, text="Help").grid(column=1, row=7, columnspan=2,
      padx=(20,0), sticky="E")
857
858   frameimg4 = ttk.Frame(frame4, padding="10 10 10 10")
```

```
859   frameimg4.grid(column=5, row=0, columnspan=2, rowspan=7, padx=(20,0))
860   frameimg4['borderwidth'] = 2
861   frameimg4['relief'] = 'groove'
862   frameimg4['width'] = 300
863   frameimg4['height'] = 350
864
865   labeles3 = ttk.Label(frame4, text='Choose rainfall input method:')
866   labeles3.grid(column=1, row=0, columnspan=4, pady=(15,0), padx=(41,42),
      sticky='N')
867   rainvar = StringVar()
868   rain = ttk.Combobox(frame4, textvariable=rainvar, state="readonly")
869   rain['values'] = ('-','T=10 Barcelona', 'IDF general equation')
870   rain.grid(column=1, row=0, columnspan=4, pady=(50,0), sticky='N')
871   rain.current(0)
872   rain.bind("<<ComboboxSelected>>", rain_method)
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887   for child in mainframe.winfo_children(): child.grid_configure(padx=5, pady=5)
888
889
890   root.mainloop()
```