

Rapport Projet Informatique

Table des matières

sujet :.....	1
graphique :.....	4
Partie HTML :.....	4
partie JavaScript :.....	5
partie brython :.....	5
autre :.....	6
Source utilisée :.....	6

sujet :

Mon sujet était le nonogramme.

Mon programme devait être capable d'en résoudre une grille.

Approche du problème :

Pour résoudre ce problème, j'ai opté pour le langage de programmation python pour la résolution et du HTML, JavaScript et brython (browser python) pour la partie graphique.

J'ai utilisé la méthode de programmation de POO permise par python.

```
# nonogramm.py > ...
1  import copy
2  from random import randint
3
4
5  > class Type: ...
8
9
10 > class Case: ...
56
57
58 > class Grille: ...
60
```

J'ai pour cela créé 3 classes, la première, Type, peut être comparée à une énumération en java, elle sert à stocker les noms de variables sans se mélanger.

La deuxième, Case, contient les données d'une classe, elle m'a servi à pouvoir avoir des variables globales pour les grilles, elle est organisée en dictionnaire dans la classe Grille.

Et la dernière classe est celle contenant toutes les cases et faisant tous les calculs nécessaires à la résolution d'un nonogramme.

C'est la classe contenant le plus de méthodes.

Chaque méthode commençant par '_' est une méthode privée n'ayant pas d'utilité en dehors de la classe.

La méthode la plus importante est la méthode remplie() qui fait appel à un moment donné à toutes les autres méthodes privées ; elle fait à elle seule le tiers du nombre de lignes du programme.

```

9
10 > class Case: ...
56
57
58 class Grille:
59
60 > def __init__(self) -> None: ...
69
70 > def copie(self): ...
77
78 > def creerGrilleHasard(self, tailleLigne, tailleColonne): ...
98
99 > def creerGrilleParIndex(self, *positions): ...
147
148 > def _positions(self, liste): ...
155
156 > def _positionParLigne(self, liste): ...
180
181 > def grilleEgal(self, grille2): ...
192
193 > def _compteTrou(self, liste): ...
212
213 > def _positionsFinal(self): ...
234
235 > def getPosition(self): ...
237
238 > def _compteTotalCase(self, type, index): ...
245
246 > def _comptePlaceLibre(self, type: str, index: int): ...
275
276 > def _verifLigneRemplis(self, liste): ...
281
282 > def _verifLimiteDebut(self, liste, indexs, compte=False): ...
308
309 > def _verifLimiteFin(self, liste, indexs, compte=False): ...
337
338 > def remplis(self, type, indexs): ...
406

```

er actif (projet informatique) Ln 193, Col 19 (10 selected) Spaces: 4 UTF-8 CRLF Python 3.11.7 64-bit (Microsoft Store)

Elle vérifie une à une tous les moyens de grappiller des valeurs dans une ligne.

```

337
338 def remplis(self, type, indexs):
339
340     coordonnee=indexs
341     liste=self.grille[type][coordonnee]
342
343     if self._verifLigneRemplis(liste):
344         return
345
346     nbLibres=self._comptePlaceLibre(type, coordonnee)
347     nbARemplir=self._compteTotalCase(type, coordonnee)
348
349     indexs=self._position[type][coordonnee]
350     trous=self._compteTrou(liste)
351
352
353     limiteDebut=self._verifLimiteDebut(liste, indexs)
354
355     try:
356         self._positionModifiable[type][coordonnee]=limiteDebut[2]
357     except:
358         pass
359
360     limiteFin=self._verifLimiteFin(liste, indexs)
361
362     try:
363         self._positionModifiable[type][coordonnee]=limiteFin[2]
364     except:
365         pass
366
367     indexsModif=self._positionModifiable[type][coordonnee]
368
369     compteur=0
370
371     if (indexsModif[0]==nbLibres[0] and indexsModif[0]==len(liste)): ...
372
373

```

```

371 > |
372 > if (indexsModif[0]==nblibres[0] and indexsModif[0]==len(liste)): ...
375 >
376 > elif indexsModif[0]==0: ...
379 >
380 >
381 >
382 > elif limiteDebut[0]: ...
390 >
391 > elif limiteFin[0]: ...
399 >
400 > elif nblibres[0]==nbAREmplir and len(indexs)>1 and not self._verifligneRemplis(liste):...
409 >
410 > elif indexsModif[nblibres[0]/2] and len(indexsModif)==1 and not self._verifligneRemplis(liste): ...
419 >
420 >
421 > elif not indexsModif and not self._verifligneRemplis(liste): ...
425 >
426 > elif indexs==self._positionPartigne(liste) and not self._verifligneRemplis(liste): ...
429 >
430 > elif len(indexs)==1 and len(self._positionPartigne(liste))>1 and not self._verifligneRemplis(liste): ...
445 >
446 > elif len(indexs)==1 and not self._verifligneRemplis(liste): ...
472 > elif max(trous)<min(indexsModif) and not self._verifligneRemplis(liste): ...
483 >
484 > def afficher(self): ...
487 >
488 > def resoud(self): ...
495 >

```

graphique :

J'ai choisi de représenter les grilles graphiques en HTML et JavaScript/brython, car je trouvais ça plus simple que de le faire grâce à des imports graphiques python que je ne connais pas.

Brython est une implémentation python 3 pour HTML 5.

Il permet la même chose que JavaScript dans une page web.

J'ai seulement utilisé du JavaScript parce que j'aime bien ce langage.

J'ai utilisé principalement le JavaScript pour créer la grille à l'écran, puis le brython pour les comportements en lien avec la classe Grille, importé depuis le fichier python présent dans le même dossier.

Partie HTML :

```
index.html M x # style.css nonogram.py
index.html > html > body > div
9 <script src="https://cdn.jsdelivr.net/npm/brython@3/brython_stdlib.js">
10 </script>
11 </head>
12 <body onload="brython()">
13 <h1>Nonogram</h1>
14 <div>
15 <p>methode de creation de la grille</p>
16 <input class="btn" type="button" value="aleatoire" id="btnAleatoire" onclick="montreZone()">
17 <input class="btn" type="button" value="index" id="btnIndex" >
18 </div>
19
20 <div id="aleatoire" style="visibility: hidden;" >
21 <p id="message">attention, toutes valeurs depassant 10 peu faire ralentir le programme</p>
22 <input type="number" name="rentrez la taille de la grille" id="entreeTaille" onkeydown="construitGrille(event)">
23 <input class="btn" type="button" id="btnValider" value="Validez" onclick="construitGrille()">
24 </div>
25
26 <table id="grille" cellpadding="0px" >
27 </table>
28
29 <div>
30 <input type="button" id="montreIndex" class="btn" value="créer les indices" style="visibility: hidden;">
31 </div>
32
33 <div style="display: flex;" style="flex-direction: column;">
34 <p id="msgApresIndice" style="visibility: hidden;">par qui voulez vous que le nonogram soit remplis ?</p>
35 </div>
36
37 <div>
38 <input type="button" value="ordinateur" class="btn" id="ordi" style="visibility: hidden;">
39 <input type="button" value="joueur" class="btn" id="joueur" style="visibility: hidden;" >
40 </div>
41 </div>
```

partie JavaScript :

```
<script>
let btnAleatoire=document.getElementById("btnAleatoire")
let btnImport=document.getElementById("btnIndex")
let entreeTaille=document.getElementById("entreeTaille")
let btnIndex=document.getElementById("montreIndex")
let grille=document.getElementById("grille")
grille.border=1

function montreZone(){
    document.getElementById("aleatoire").style.visibility="visible"
}

function construitGrille(event){
    if (levent || event.key=="Enter"){
        while (grille.rows.length > 0) {
            grille.deleteRow(0)
        }
        btnIndex.style.visibility="hidden"
        if (entreeTaille.value>0){
            btnIndex.style="visible"
            for (i=0;i<entreeTaille.value;i++){
                var ligne=grille.insertRow()
                for (j=0;j<entreeTaille.value;j++){
                    var cellule=ligne.insertCell()
                    cellule.style.padding="10px"
                    cellule.id=i+'_'+j
                }
            }
        }
    }
}

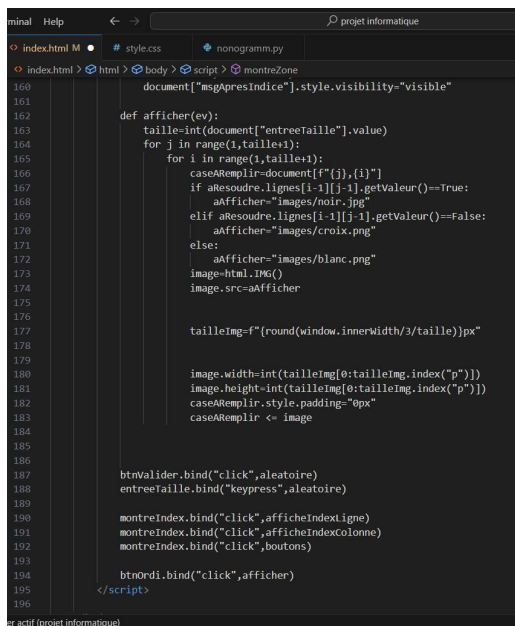
</script>
```

partie brython :

```
99
100
101 <script type="text/python">
102     from browser import document
103     from browser import html
104     from browser import window
105     from nonogram import Grille, Type
106
107     btnValider=document["btnValider"]
108     entreeTaille=document["entreeTaille"]
109     tableau=document["grille"]
110     aResoudre=Grille()
111     resolu=Grille()
112     idPremiereLigne=[]
113     idPremiereColonne=[]
114     montreIndex=document["montreIndex"]
115     btnOrdi=document["ordi"]
116     btnJoueur=document["joueur"]
117
118     def aleatoire(ev):
119         if ev.type=="click":
120             ev.key="Enter"
121         if ev.key=="Enter" or ev.type=="click":
122             taille=document["entreeTaille"].value
123             if taille=="":taille=0
124             taille=int(taille)
125
126             if taille>0:
127
128                 for i in range(taille+1):
129                     idPremiereLigne.append(f"0,{i}")
130                     idPremiereColonne.append(f"{i},0")
131
132                 aResoudre.creerGrilleHasard(taille,taille)
133                 resolu.lignes=aResoudre.copie()
134                 aResoudre.resoud()
135
```

chier actif (projet informatique)

```
terminal Help ← → projet informatique
index.html M ● # style.css nonogram.py
index.html > html > body > script > montreZone
135
136     def afficheIndexLigne(ev):
137         Index=aResoudre.getPosition()[Type.ligne]
138
139         for i in idPremiereLigne[1:]:
140             celluleARemplir=document[i]
141             valeur=l.split(",")
142             chaine=index[int(valeur[1])-1]
143             chaine=html.P("<br>".join(map(str,chaine)))
144             document[i] <= chaine
145
146     def afficheIndexColonne(ev):
147         Index=aResoudre.getPosition()[Type.colonne]
148
149         for i in idPremiereColonne[1:]:
150             celluleARemplir=document[i]
151             valeur=i.split(",")
152             chaine=index[int(valeur[0])-1]
153             chaine=html.P(" ".join(map(str,chaine)))
154             document[i] <= chaine
155
156     def boutons(ev):
157         montreIndex.style.visibility="hidden"
158         btnOrdi.style.visibility="visible"
159         btnJoueur.style.visibility="visible"
160         document["msgApresIndex"].style.visibility="visible"
161
162     def afficher(ev):
163         taille=int(document["entreeTaille"].value)
164         for j in range(1,taille+1):
165             for i in range(1,taille+1):
166                 caseARemplir=document[f"{j},{i}"]
167                 if aResoudre.lignes[i-1][j-1].getValeur()==True:
168                     aAfficher="images/noir.jpg"
169                 elif aResoudre.lignes[i-1][j-1].getValeur()==False:
170                     aAfficher="images/croix.png"
171                 else:
```



```
160 document["msgApresIndice"].style.visibility="visible"
161
162 def afficher(v):
163     taille=Int(document["entreeTaille"].value)
164     for j in range(1,taille+1):
165         for i in range(1,taille+1):
166             caseARemplir=document["f"+j+i]
167             if aResoudre.lignes[i-1][j-1].getValeur()==True:
168                 aAfficher="images/noir.jpg"
169             elif aResoudre.lignes[i-1][j-1].getValeur()==False:
170                 aAfficher="images/croix.png"
171             else:
172                 aAfficher="images/blanc.png"
173             image=html.IWG()
174             image.src=aAfficher
175
176             tailleImg=f"(round(window.innerWidth/3/taille))px"
177
178             image.width=Int(tailleImg[0:tailleImg.index("p")])
179             image.height=Int(tailleImg[0:tailleImg.index("p")])
180             caseARemplir.style.padding="0px"
181             caseARemplir <= image
182
183
184
185
186 btnValider.bind("click",aleatoire)
187 entreeTaille.bind("keypress",aleatoire)
188
189
190 montreIndex.bind("click",afficheIndexligne)
191 montreIndex.bind("click",afficheIndexcolonne)
192 montreIndex.bind("click",boutons)
193
194 btnOrdre.bind("click",afficher)
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

autre :

J'ai aussi utilisé un Git pour pouvoir revenir en arrière en cas de mauvaise manipulation de ma part. Avec un remote sur GitHub, d'ailleurs ce fichier étant dans le même dossier que le code, il fait aussi partie du Git.

[lien du Git](#)

Source utilisée :

Merlin et ChatGPT (comme documentation et recherche d'erreur)

<https://brython.info/>

(pour les fonctions propres à brython)