

Coursework 1 Report

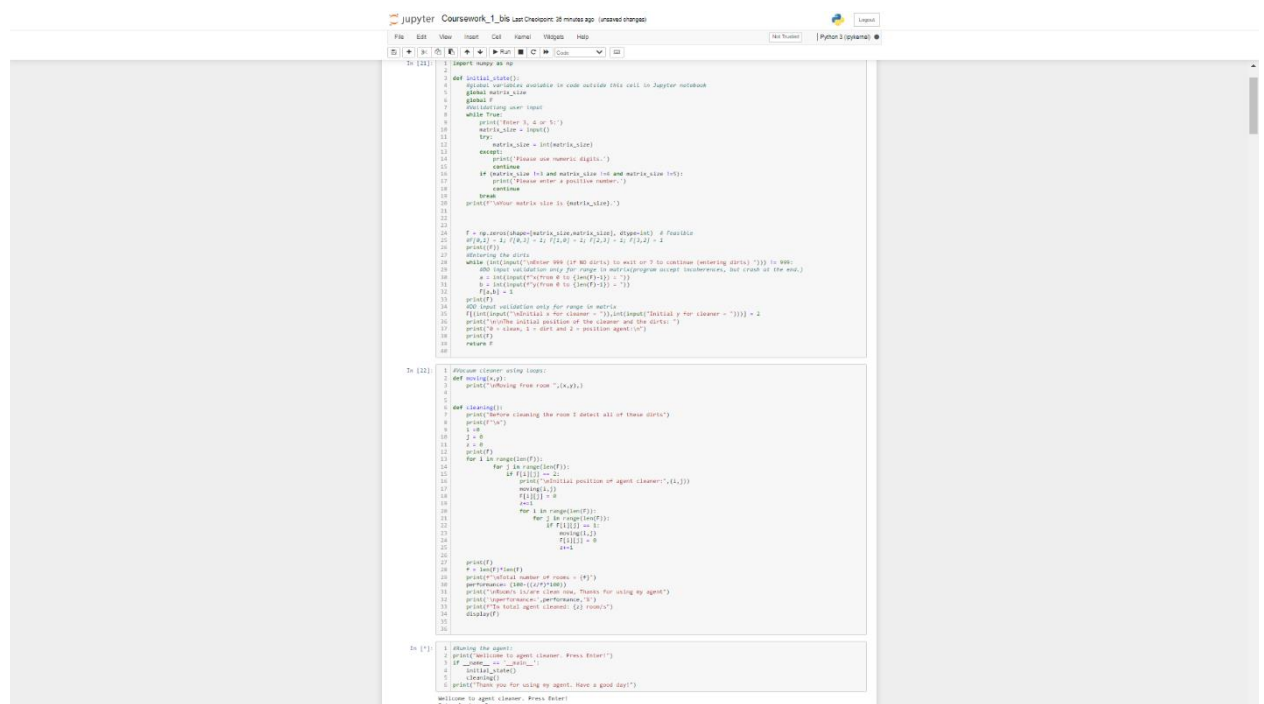
YIT19488399

Delivery date: 15 February 2022

Value: 25% of total

For designing the program of vacuum cleaner, I used and followed instructions from “Coursework 1 Specifications”. I used Python 3 (programming language) and IDE was Jupyter Notebook. I used my computer and I worked in class during lab time, at the University of Roehampton’s library, and in my student room. I asked classmates about any idea how to do the coursework and answers were very vague. Following the recommendations of the module about reading, I checked both books online and find both books very useful, especially: “Artificial Intelligence: a Modern Approach, EBook, Global Edition: A Modern Approach” by Stuart Russell and Peter Norvig; pages 36, 70 and 143. The second book was: “Introduction to Artificial Intelligence” by Wolfgang Ertel and Nathanael T. Black; was less useful for this Coursework 1, but helped me to understand the logic of my vacuum cleaner design.

My coursework has 3 vacuum cleaner solutions. Starting from a simple and easy one to the final one and more elaborated and professional. The final solution can be improved by optimizing the movement of the cleaner in the matrix (this would be 4th solution). Solution number 1 is very simple. I used reference: <https://github.com/A-safarij/vacuum-cleaner-agent/blob/main/vacuum-cleaner.py>. Images follow:



```
In [121]: 1 import random as np
2 def initial_state():
3     """Returns a random available to code inside this cell to Jupyter notebook
4     global matrix_size
5     global F
6     #Initializing user input
7     while True:
8         print('Enter 3, 4 or 5:')
9         matrix_size = input()
10        try:
11            matrix_size = int(matrix_size)
12        except:
13            print('Please use numeric digits.')
14        continue
15        if (matrix_size != 3 and matrix_size != 4 and matrix_size != 5):
16            print('Please enter a positive number.')
17        continue
18        break
19    print('Your matrix size is (matrix_size).')
20
21 F = np.zeros(shape=(matrix_size,matrix_size), dtype=int) # Function
22 def clean():
23     for (x,y) in [(0,0), (1,0), (1,1), (2,1), (2,2)]:
24         F[x,y] = 1
25     print(F)
26
27 #Showing the dirt
28 while (int(input('Enter 00 (if 00 dirt) to exit or 0 to continue (entering dirt)')) != 000):
29     #do input validation only for range to matrix(program accept numbers, but crash at the end.)
30     x = int(input('Enter 0 to (matrix_x)-1'))
31     y = int(input('Enter 0 to (matrix_y)-1'))
32     F[x,y] = 1
33     print(F)
34
35 #do input validation only for range in matrix
36 if (int(input('Initial x for cleaner = ')) < 0 or int(input('Initial y for cleaner = ')) < 0):
37     print('x or y is the initial position of the cleaner and the dirt.')
38     print('x = clean, 1 = dirt and 2 = position agent(x)')
39     print(F)
40     return F
41
42 In [122]: 1 #vacuum cleaner using loops
2 def moving(x,y):
3     print('moving from row "(x,y)"')
4
5 def cleaning():
6     print('Before cleaning the row 1 detect all of these dirt')
7     print(F)
8     x = 0
9     y = 0
10    z = 0
11    for i in range(len(F)):
12        for j in range(len(F)):
13            if F[i][j] == 1:
14                #print('Initial position of agent cleaner',(i,j))
15                moving(i,j)
16                F[i][j] = 0
17            else:
18                for i in range(len(F)):
19                    for j in range(len(F)):
20                        if F[i][j] == 1:
21                            moving(i,j)
22                            F[i][j] = 0
23                            break
24
25    print(F)
26    # = len(F)*len(F)
27    #performance: len(F)*len(F)
28    print('Thanks for using my agent')
29    print('agent performance: performance: 0')
30    print('The total agent cleaned: (z) row(s)')
31    display(F)
32
33 In [2]: 1 #Showing the agent:
2 def print_state_to_agent_cleaner:
3     if name == '__main__':
4         initial_state()
5         cleaning()
6     print('Thank you for using my agent. Have a good day!')
7
8 welcome to agent cleaner. Press Enter
9 Enter 3, 4 or 5.
```

```

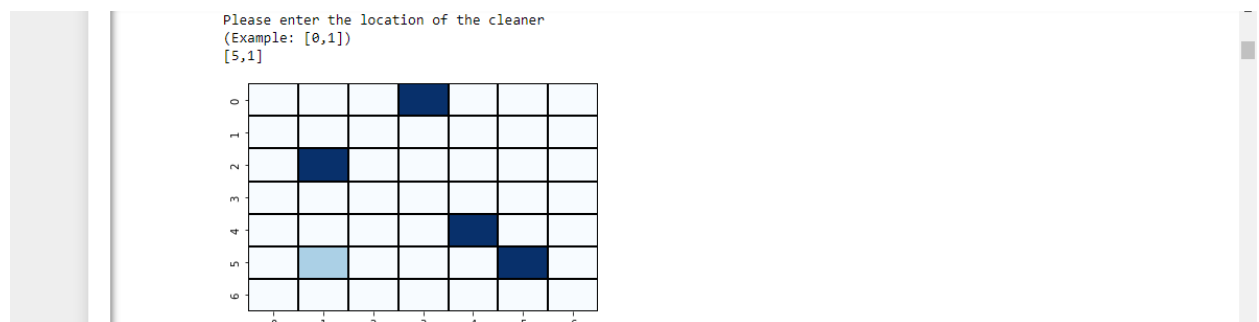
jupyter Coursework_1_bis Last Checkpoint 37 minutes ago (unsaved changes)
File Edit View Insert Cell Help Run Code
In [16]:
1 #offering the agent:
2 print("Welcome to agent cleaner. Press Enter!")
3 if __name__ == "__main__":
4     #initial_conditions
5     #cleaning()
6     print("Thank you for using my agent. Have a good day!")
7
8 Welcome to agent cleaner. Press Enter!
9 Enter 3, 4 or 5:
10
11 Please enter numeric digits.
12 Enter 3, 4 or 5:
13
14 Please enter a positive number.
15 Enter 3, 4 or 5:
16 4
17
18 Your matrix size is 4.
19 [[0 0 0]
20  [0 0 0]
21  [0 0 0]
22  [0 0 0]]
23
24 Enter 000 (if NO dirt) to exit or 7 to continue (entering dirt) ?
25 x\from 0 to 3: 2
26 y\from 0 to 3: 1
27
28 Enter 000 (if NO dirt) to exit or 7 to continue (entering dirt) ?
29 x\from 0 to 3: 0
30 y\from 0 to 3: 2
31
32 Enter 000 (if NO dirt) to exit or 7 to continue (entering dirt) 000
33 [[0 0 0]
34  [0 0 0]
35  [0 0 0]
36  [0 0 0]]
37
38 Initial x for cleaner = 1
39 Initial y for cleaner = 2
40
41 The initial position of the cleaner and the dirt:
42 x = clean, 1 = dirt and 2 = position agent:
43
44 [[0 0 0]
45  [0 0 2]
46  [0 0 0]
47  [0 0 0]]
48
49 Before cleaning the room I detect all of these dirt
50
51 [[0 0 0]
52  [0 0 2]
53  [0 0 0]
54  [0 0 0]]
55
56 Initial position of agent cleaner: (1, 2)
57
58 Moving from room (1, 2)
59
60 Moving from room (0, 2)
61 [[0 0 0]
62  [0 0 0]
63  [0 0 0]
64  [0 0 0]]
65
66 Total number of rooms = 16
67 Room's is/care clean now, Thanks for using my agent
68 performance: 61.25 %
69 In total agent cleaned: 3 room/s
70 array([[0, 0, 0],
71       [0, 0, 0],
72       [0, 0, 0],
73       [0, 0, 0]])
74
75 Thank you for using my agent. Have a good day!
76
77 In [17]:
78 1 #####
79 2 #####
80 3 #####
81 4 #####
82 5 #####
83 6 #####
84 7 #####
85 8 #####
86 9 #####
87 10 #####
88 11 #####
89 12 #####
90 13 #####
91 14 #####
92 15 #####
93 16 #####
94 17 #####
95 18 #####
96 19 #####
97 20 #####
98 21 #####
99 22 #####
100 23 #####
101 24 #####
102 25 #####
103 26 #####
104 27 #####
105 28 #####
106 29 #####
107 30 #####
108 31 #####
109 32 #####
110 33 #####
111 34 #####
112 35 #####
113 36 #####
114 37 #####
115 38 #####
116 39 #####
117 40 #####
118 41 #####
119 42 #####
120 43 #####
121 44 #####
122 45 #####
123 46 #####
124 47 #####
125 48 #####
126 49 #####
127 50 #####
128 51 #####
129 52 #####
130 53 #####
131 54 #####
132 55 #####
133 56 #####
134 57 #####
135 58 #####
136 59 #####
137 60 #####
138 61 #####
139 62 #####
140 63 #####
141 64 #####
142 65 #####
143 66 #####
144 67 #####
145 68 #####
146 69 #####
147 70 #####
148 71 #####
149 72 #####
150 73 #####
151 74 #####
152 75 #####
153 76 #####
154 77 #####
155 78 #####
156 79 #####
157 80 #####
158 81 #####
159 82 #####
160 83 #####
161 84 #####
162 85 #####
163 86 #####
164 87 #####
165 88 #####
166 89 #####
167 90 #####
168 91 #####
169 92 #####
170 93 #####
171 94 #####
172 95 #####
173 96 #####
174 97 #####
175 98 #####
176 99 #####
177 100 #####
178 101 #####
179 102 #####
180 103 #####
181 104 #####
182 105 #####
183 106 #####
184 107 #####
185 108 #####
186 109 #####
187 110 #####
188 111 #####
189 112 #####
190 113 #####
191 114 #####
192 115 #####
193 116 #####
194 117 #####
195 118 #####
196 119 #####
197 120 #####
198 121 #####
199 122 #####
200 123 #####
201 124 #####
202 125 #####
203 126 #####
204 127 #####
205 128 #####
206 129 #####
207 130 #####
208 131 #####
209 132 #####
210 133 #####
211 134 #####
212 135 #####
213 136 #####
214 137 #####
215 138 #####
216 139 #####
217 140 #####
218 141 #####
219 142 #####
220 143 #####
221 144 #####
222 145 #####
223 146 #####
224 147 #####
225 148 #####
226 149 #####
227 150 #####
228 151 #####
229 152 #####
230 153 #####
231 154 #####
232 155 #####
233 156 #####
234 157 #####
235 158 #####
236 159 #####
237 160 #####
238 161 #####
239 162 #####
240 163 #####
241 164 #####
242 165 #####
243 166 #####
244 167 #####
245 168 #####
246 169 #####
247 170 #####
248 171 #####
249 172 #####
250 173 #####
251 174 #####
252 175 #####
253 176 #####
254 177 #####
255 178 #####
256 179 #####
257 180 #####
258 181 #####
259 182 #####
260 183 #####
261 184 #####
262 185 #####
263 186 #####
264 187 #####
265 188 #####
266 189 #####
267 190 #####
268 191 #####
269 192 #####
270 193 #####
271 194 #####
272 195 #####
273 196 #####
274 197 #####
275 198 #####
276 199 #####
277 200 #####
278 201 #####
279 202 #####
280 203 #####
281 204 #####
282 205 #####
283 206 #####
284 207 #####
285 208 #####
286 209 #####
287 210 #####
288 211 #####
289 212 #####
290 213 #####
291 214 #####
292 215 #####
293 216 #####
294 217 #####
295 218 #####
296 219 #####
297 220 #####
298 221 #####
299 222 #####
300 223 #####
301 224 #####
302 225 #####
303 226 #####
304 227 #####
305 228 #####
306 229 #####
307 230 #####
308 231 #####
309 232 #####
310 233 #####
311 234 #####
312 235 #####
313 236 #####
314 237 #####
315 238 #####
316 239 #####
317 240 #####
318 241 #####
319 242 #####
320 243 #####
321 244 #####
322 245 #####
323 246 #####
324 247 #####
325 248 #####
326 249 #####
327 250 #####
328 251 #####
329 252 #####
330 253 #####
331 254 #####
332 255 #####
333 256 #####
334 257 #####
335 258 #####
336 259 #####
337 260 #####
338 261 #####
339 262 #####
340 263 #####
341 264 #####
342 265 #####
343 266 #####
344 267 #####
345 268 #####
346 269 #####
347 270 #####
348 271 #####
349 272 #####
350 273 #####
351 274 #####
352 275 #####
353 276 #####
354 277 #####
355 278 #####
356 279 #####
357 280 #####
358 281 #####
359 282 #####
360 283 #####
361 284 #####
362 285 #####
363 286 #####
364 287 #####
365 288 #####
366 289 #####
367 290 #####
368 291 #####
369 292 #####
370 293 #####
371 294 #####
372 295 #####
373 296 #####
374 297 #####
375 298 #####
376 299 #####
377 300 #####
378 301 #####
379 302 #####
380 303 #####
381 304 #####
382 305 #####
383 306 #####
384 307 #####
385 308 #####
386 309 #####
387 310 #####
388 311 #####
389 312 #####
390 313 #####
391 314 #####
392 315 #####
393 316 #####
394 317 #####
395 318 #####
396 319 #####
397 320 #####
398 321 #####
399 322 #####
400 323 #####
401 324 #####
402 325 #####
403 326 #####
404 327 #####
405 328 #####
406 329 #####
407 330 #####
408 331 #####
409 332 #####
410 333 #####
411 334 #####
412 335 #####
413 336 #####
414 337 #####
415 338 #####
416 339 #####
417 340 #####
418 341 #####
419 342 #####
420 343 #####
421 344 #####
422 345 #####
423 346 #####
424 347 #####
425 348 #####
426 349 #####
427 350 #####
428 351 #####
429 352 #####
430 353 #####
431 354 #####
432 355 #####
433 356 #####
434 357 #####
435 358 #####
436 359 #####
437 360 #####
438 361 #####
439 362 #####
440 363 #####
441 364 #####
442 365 #####
443 366 #####
444 367 #####
445 368 #####
446 369 #####
447 370 #####
448 371 #####
449 372 #####
450 373 #####
451 374 #####
452 375 #####
453 376 #####
454 377 #####
455 378 #####
456 379 #####
457 380 #####
458 381 #####
459 382 #####
460 383 #####
461 384 #####
462 385 #####
463 386 #####
464 387 #####
465 388 #####
466 389 #####
467 390 #####
468 391 #####
469 392 #####
470 393 #####
471 394 #####
472 395 #####
473 396 #####
474 397 #####
475 398 #####
476 399 #####
477 400 #####
478 401 #####
479 402 #####
480 403 #####
481 404 #####
482 405 #####
483 406 #####
484 407 #####
485 408 #####
486 409 #####
487 410 #####
488 411 #####
489 412 #####
490 413 #####
491 414 #####
492 415 #####
493 416 #####
494 417 #####
495 418 #####
496 419 #####
497 420 #####
498 421 #####
499 422 #####
500 423 #####
501 424 #####
502 425 #####
503 426 #####
504 427 #####
505 428 #####
506 429 #####
507 430 #####
508 431 #####
509 432 #####
510 433 #####
511 434 #####
512 435 #####
513 436 #####
514 437 #####
515 438 #####
516 439 #####
517 440 #####
518 441 #####
519 442 #####
520 443 #####
521 444 #####
522 445 #####
523 446 #####
524 447 #####
525 448 #####
526 449 #####
527 450 #####
528 451 #####
529 452 #####
530 453 #####
531 454 #####
532 455 #####
533 456 #####
534 457 #####
535 458 #####
536 459 #####
537 460 #####
538 461 #####
539 462 #####
540 463 #####
541 464 #####
542 465 #####
543 466 #####
544 467 #####
545 468 #####
546 469 #####
547 470 #####
548 471 #####
549 472 #####
550 473 #####
551 474 #####
552 475 #####
553 476 #####
554 477 #####
555 478 #####
556 479 #####
557 480 #####
558 481 #####
559 482 #####
560 483 #####
561 484 #####
562 485 #####
563 486 #####
564 487 #####
565 488 #####
566 489 #####
567 490 #####
568 491 #####
569 492 #####
570 493 #####
571 494 #####
572 495 #####
573 496 #####
574 497 #####
575 498 #####
576 499 #####
577 500 #####
578 501 #####
579 502 #####
580 503 #####
581 504 #####
582 505 #####
583 506 #####
584 507 #####
585 508 #####
586 509 #####
587 510 #####
588 511 #####
589 512 #####
590 513 #####
591 514 #####
592 515 #####
593 516 #####
594 517 #####
595 518 #####
596 519 #####
597 520 #####
598 521 #####
599 522 #####
600 523 #####
601 524 #####
602 525 #####
603 526 #####
604 527 #####
605 528 #####
606 529 #####
607 530 #####
608 531 #####
609 532 #####
610 533 #####
611 534 #####
612 535 #####
613 536 #####
614 537 #####
615 538 #####
616 539 #####
617 540 #####
618 541 #####
619 542 #####
620 543 #####
621 544 #####
622 545 #####
623 546 #####
624 547 #####
625 548 #####
626 549 #####
627 550 #####
628 551 #####
629 552 #####
630 553 #####
631 554 #####
632 555 #####
633 556 #####
634 557 #####
635 558 #####
636 559 #####
637 560 #####
638 561 #####
639 562 #####
640 563 #####
641 564 #####
642 565 #####
643 566 #####
644 567 #####
645 568 #####
646 569 #####
647 570 #####
648 571 #####
649 572 #####
650 573 #####
651 574 #####
652 575 #####
653 576 #####
654 577 #####
655 578 #####
656 579 #####
657 580 #####
658 581 #####
659 582 #####
660 583 #####
661 584 #####
662 585 #####
663 586 #####
664 587 #####
665 588 #####
666 589 #####
667 590 #####
668 591 #####
669 592 #####
670 593 #####
671 594 #####
672 595 #####
673 596 #####
674 597 #####
675 598 #####
676 599 #####
677 600 #####
678 601 #####
679 602 #####
680 603 #####
681 604 #####
682 605 #####
683 606 #####
684 607 #####
685 608 #####
686 609 #####
687 610 #####
688 611 #####
689 612 #####
690 613 #####
691 614 #####
692 615 #####
693 616 #####
694 617 #####
695 618 #####
696 619 #####
697 620 #####
698 621 #####
699 622 #####
700 623 #####
701 624 #####
702 625 #####
703 626 #####
704 627 #####
705 628 #####
706 629 #####
707 630 #####
708 631 #####
709 632 #####
710 633 #####
711 634 #####
712 635 #####
713 636 #####
714 637 #####
715 638 #####
716 639 #####
717 640 #####
718 641 #####
719 642 #####
720 643 #####
721 644 #####
722 645 #####
723 646 #####
724 647 #####
725 648 #####
726 649 #####
727 650 #####
728 651 #####
729 652 #####
730 653 #####
731 654 #####
732 655 #####
733 656 #####
734 657 #####
735 658 #####
736 659 #####
737 660 #####
738 661 #####
739 662 #####
740 663 #####
741 664 #####
742 665 #####
743 666 #####
744 667 #####
745 668 #####
746 669 #####
747 670 #####
748 671 #####
749 672 #####
750 673 #####
751 674 #####
752 675 #####
753 676 #####
754 677 #####
755 678 #####
756 679 #####
757 680 #####
758 681 #####
759 682 #####
760 683 #####
761 684 #####
762 685 #####
763 686 #####
764 687 #####
765 688 #####
766 689 #####
767 690 #####
768 691 #####
769 692 #####
770 693 #####
771 694 #####
772 695 #####
773 696 #####
774 697 #####
775 698 #####
776 699 #####
777 700 #####
778 701 #####
779 702 #####
780 703 #####
781 704 #####
782 705 #####
783 706 #####
784 707 #####
785 708 #####
786 709 #####
787 710 #####
788 711 #####
789 712 #####
790 713 #####
791 714 #####
792 715 #####
793 716 #####
794 717 #####
795 718 #####
796 719 #####
797 720 #####
798 721 #####
799 722 #####
800 723 #####
801 724 #####
802 725 #####
803 726 #####
804 727 #####
805 728 #####
806 729 #####
807 730 #####
808 731 #####
809 732 #####
810 733 #####
811 734 #####
812 735 #####
813 736 #####
814 737 #####
815 738 #####
816 739 #####
817 740 #####
818 741 #####
819 742 #####
820 743 #####
821 744 #####
822 745 #####
823 746 #####
824 747 #####
825 748 #####
826 749 #####
827 750 #####
828 751 #####
829 752 #####
830 753 #####
831 754 #####
832 755 #####
833 756 #####
834 757 #####
835 758 #####
836 759 #####
837 760 #####
838 761 #####
839 762 #####
840 763 #####
841 764 #####
842 765 #####
843 766 #####
844 767 #####
845 768 #####
846 769 #####
847 770 #####
848 771 #####
849 772 #####
850 773 #####
851 774 #####
852 775 #####
853 776 #####
854 777 #####
855 778 #####
856 779 #####
857 780 #####
858 781 #####
859 782 #####
860 783 #####
861 784 #####
862 785 #####
863 786 #####
864 787 #####
865 788 #####
866 789 #####
867 790 #####
868 791 #####
869 792 #####
870 793 #####
871 794 #####
872 795 #####
873 796 #####
874 797 #####
875 798 #####
876 799 #####
877 800 #####
878 801 #####
879 802 #####
880 803 #####
881 804 #####
882 805 #####
883 806 #####
884 807 #####
885 808 #####
886 809 #####
887 810 #####
888 811 #####
889 812 #####
890 813 #####
891 814 #####
892 815 #####
893 816 #####
894 817 #####
895 818 #####
896 819 #####
897 820 #####
898 821 #####
899 822 #####
900 823 #####
901 824 #####
902 825 #####
903 826 #####
904 827 #####
905 828 #####
906 829 #####
907 830 #####
908 831 #####
909 832 #####
910 833 #####
911 834 #####
912 835 #####
913 836 #####
914 837 #####
915 838 #####
916 839 #####
917 840 #####
918 841 #####
919 842 #####
920 843 #####
921 844 #####
922 845 #####
923 846 #####
924 847 #####
925 848 #####
926 849 #####
927 850 #####
928 851 #####
929 852 #####
930 853 #####
931 854 #####
932 855 #####
933 856 #####
934 857 #####
935 858 #####
936 859 #####
937 860 #####
938 861 #####
939 862 #####
940 863 #####
941 864 #####
942 865 #####
943 866 #####
944 867 #####
945 868 #####
946 869 #####
947 870 #####
948 871 #####
949 872 #####
950 873 #####
951 874 #####
952 875 #####
953 876 #####
954 877 #####
955 878 #####
956 879 #####
957 880 #####
958 881 #####
959 882 #####
960 883 #####
961 884 #####
962 885 #####
963 886 #####
964 887 #####
965 888 #####
966 889 #####
967 890 #####
968 891 #####
969 892 #####
970 893 #####
971 894 #####
972 895 #####
973 896 #####
974 897 #####
975 898 #####
976 899 #####
977 900 #####
978 901 #####
979 902 #####
980 903 #####
981 904 #####
982 905 #####
983 906 #####
984 907 #####
985 908 #####
986 909 #####
987 910 #####
988 911 #####
989 912 #####
990 913 #####
991 914 #####
992 915 #####
993 916 #####
994 917 #####
995 918 #####
996 919 #####
997 920 #####
998 921 #####
999 922 #####
1000 923 #####
1001 924 #####
1002 925 #####
1003 926 #####
1004 927 #####
1005 928 #####
1006 929 #####
1007 930 #####
1008 931 #####
1009 932 #####
1010 933 #####
1011 934 #####
1012 935 #####
1013 936 #####
1014 937 #####
1015 938 #####
1016 939 #####
1017 940 #####
1018 941 #####
1019 942 #####
1020 943 #####
1021 944 #####
1022 945 #####
1023 946 #####
1024 947 #####
1025 948 #####
1026 949 #####
1027 950 #####
1028 951 #####
1029 952 #####
1030 953 #####
1031 954 #####
1032 955 #####
1033 956 #####
1034 957 #####
1035 958 #####
1036 959 #####
1037 960 #####
1038 961 #####
1039 962 #####
1040 963 #####
1041 964 #####
1042 965 #####
1043 966 #####
1044 967 #####
1045 968 #####
1046 969 #####
1047 970 #####
1048 971 #####
1049 972 #####
1050 973 #####
1051 974 #####
1052 975 #####
1053 976 #####
1054 977 #####
1055 978 #####
1056 979 #####
1057 980 #####
1058 981 #####
1059 982 #####
1060 983 #####
1061 984 #####
1062 985 #####
1063 986 #####
1064 987 #####
1065 988 #####
1066 989 #####
1067 990 #####
1068 991 #####
1069 992 #####
1070 993 #####
1071 994 #####
1072 995 #####
1073 996 #####
1074 997 #####
1075 998 #####
1076 999 #####
1077 1000 #####
1078 1001 #####
1079 1002 #####
1080 1003 #####
1081 1004 #####
1082 1005 #####
1083 1006 #####
1084 1007 #####
1085 1008 #####
1086 1009 #####
1087 1010 #####
1088 1011 #####
1089 1012 #####
1090 1013 #####
1091 1014 #####
1092 1015 #####
1093 1016 #####
1094 1017 #####
1095 1018 #####
1096 1019 #####
1097 1020 #####
1098 1021 #####
1099 1022 #####
1100 1023 #####
1101 1024 #####
1102 1025 #####
1103 1026 #####
1104 1027 #####
1105 1028 #####
1106 1029 #####
1107 1030 #####
1108 1031 #####
1109 1032 #####
1110 1033 #####
1111 1034 #####
1112 1035 #####
1113 1036 #####
1114 1037 #####
1115 1038 #####
1116 1039 #####
1117 1040 #####
1118 1041 #####
1119 1042 #####
1120 1043 #####
1121 1044 #####
1122 1045 #####
1123 1046 #####
1124 1047 #####
1125 1048 #####
1126 1049 #####
1127 1050 #####
1128 1051 #####
1129 1052 #####
1130 1053 #####
1131 1054 #####
1132 1055 #####
1133 1056 #####
1134 1057 #####
1135 1058 #####
1136 1059 #####
1137 1060 #####
1138 1061 #####
1139 1062 #####
1140 1063 #####
1141 1064 #####
1142 1065 #####
1143 1066 #####
1144 1067 #####
1145 1068 #####
1146 1069 #####
1147 1070 #####
1148 1071 #####
1149 1072 #####
1150 1073 #####
1151 1074 #####
1152 1075 #####
1153 1076 #####
1154 1077 #####
1155 1078 #####
1156 1079 #####
1157 1080 #####
1158 1081 #####
1159 1082 #####
1160 1083 #####
1161 1084 #####
1162 1085 #####
1163 1086 #####
1164 1087 #####
1165 1088 #####
1166 1089 #####
1167 1090 #####
1168 1091 #####
1169 1092 #####
1170 1093 #####
1171 1094 #####
1172 1095 #####
1173 1096 #####
1174 1097 #####
1175 1098 #####
1176 1099 #####
1177 1100 #####
1178 1101 #####
1179 1102 #####
1180 1103 #####
1181 1104 #####
1182 1105 #####
1183 1106 #####
1184 1107 #####
1185 1108 #####
1186 1109 #####
1187 1110 #####
1188 1111 #####
1189 1112 #####
1190 1113 #####
1191 1114 #####
1192 1115 #####
1193 1116 #####
1194 1117 #####
1195 1118 #####
1196 1119 #####
1197 1120 #####
1198 1121 #####
1199 1122 #####
1200 1123 #####
1201 1124 #####
1202 1125 #####
1203 1126 #####
1204 1127 #####
1205 1128 #####
1206 1129 #####
1207 1130 #####
1208 1131 #####
1209 1132 #####
1210 1133 #####
1211 1134 #####
1212 1135 #####
1213 1136 #####
1214 1137 #####
1215 1138 #####
1216 1139 #####
1217 1140 #####
1218 1141 #####
1219 1142 #####
1220 1143 #####
1221 1144 #####
1222 1145 #####
1223 1146 #####
1224 1147 #####
1225 1148 #####
1226 1149 #####
1227 1150 #####
1228 1151 #####
1229 1152 #####
1230 1153 #####
1231 1154 #####
1232 1155 #####
1233 1156 #####
1234 1157 #####
1235 1158 #####
1236 1159 #####
1237 1160 #####
1238 1161 #####
1239 1162 #####
1240 1163 #####
1241 1164 #####
1242 1165 #####
1243 1166 #####
1244 1167 #####
1245 1168 #####
1246 1169 #####
1247 1170 #####
1248 1171 #####
1249 1172 #####
1250 1173 #####
1251 1174 #####
1252 1175 #####
1253 1176 #####
1254 1177 #####
1255 1178 #####
1256 1179 #####
1257 1180 #####
1258 1181 #####
1259 1182 #####
1260 1183 #####
1261 1184 #####
1262 1185 #####
1263 1186 #####
1264 1187 #####
1265 1188 #####
1266 1189 #####
1267 1190 #####
1268 1191 #####
1269 1192 #####
1270 1193 #####
1271 1194 #####
1272 1195 #####
1273 1196 #####
1274 1197 #####
1275 1198 #####
1276 1199 #####
1277 1200 #####
1278 1201 #####
1279 1202 #####
1280 1203 #####
1281 1204 #####
1282 1205 #####
1283 1206 #####
1284 1207 #####
1285 1208 #####
1286 1209 #####
1287 1210 #####
1288 1211 #####
1289 1212 #####
1290 1213 #####
1291 1214 #####
1292 1215 #####
1293 1216 #####
1294 1217 #####
1295 1218 #####
1296 1219 #####
1297 1220 #####
1298 1221 #####
1299 1222 #####
1300 1223 #####
1301 1224 #####
1302 1225 #####
1303 1226 #####
1304 1227 #####
1305 1228 #####
1306 1229 #####
1307 1230 #####
1308 1231 #####
1309 1232 #####
1310 1233 #####
1311 1234 #####
1312 1235 #####
1313 12
```

Solution number 3 follows logic from solution Lab2 and previous reference from the Internet (<https://notebook.community/pblanc5/Artificial-Intelligence-/Homework2/Homework%202>). It is a simple algorithm and easy to understand. But because of the use of many functions (13 in total) plus driving block (last cell of code) and importing libraries with initialisation of variables (2 Jupyter notebook code cells in one); it is very easy to get lost in the code. The total number of cells is 15. For a computer is easy because it retains in memory all functions with corresponding variables. While programming, it is much easier to divide all the code into small pieces and this is what I did. While loops are designed in a way that never my program will do an infinite loop. Many comments are included in the code to help to understand my program because it jumps from one block to another frequently. I believe that will be around 1800 steps executed in my code for a simple matrix of size 3x3. Time and space complexity I think increase exponentially when bigger size of the matrix like 5x5. Lab2 solution has about 370 steps for a line of 9 cells in the horizontal position. Reference:

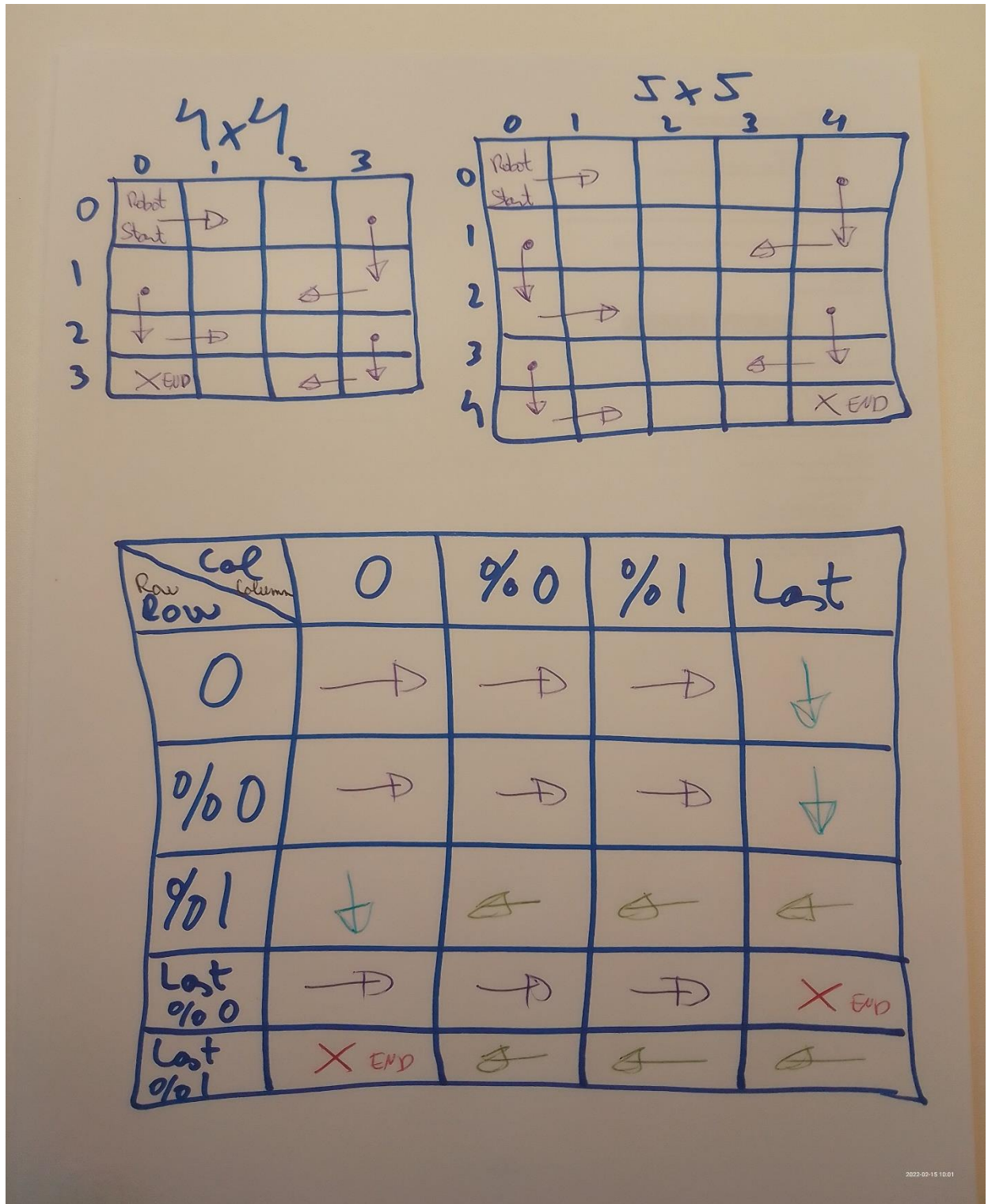
<https://pythontutor.com/visualize.html#mode=edit>.

Solution number 3 ask the user to input the size of the matrix. A bug that I leave in code is the possibility to enter any size because is very exciting to see how the agent is cleaning. After, I ask the user for input dirt coordinates. Same procedure to input location of the agent at the beginning of the program. Before starting my agent to clean, I display nicely on the screen to the user the map of the initial situation.

Resource used for map: <https://seaborn.pydata.org/generated/seaborn.heatmap.html>. All previous steps are verified by function "input_row_col()" and regex: <https://regexone.com/references/python>. The map will display dark blue colour for specks of dirt (3), less dark blue colour when the agent is on dirt (2), and the colour of the agent is light blue (1). White/Transparent is when the cell of the matrix is clean (0). At the end of the cleaning of the agent, the matrix only has one light blue colour (1), because the agent is in the matrix. Agent stop to move and program finish. The reason why my map is composed of rectangular cells is because of the use of the seaborn library (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>). My map now looks much better than the previous solution (Solution 2) using the matplotlib library. Image:

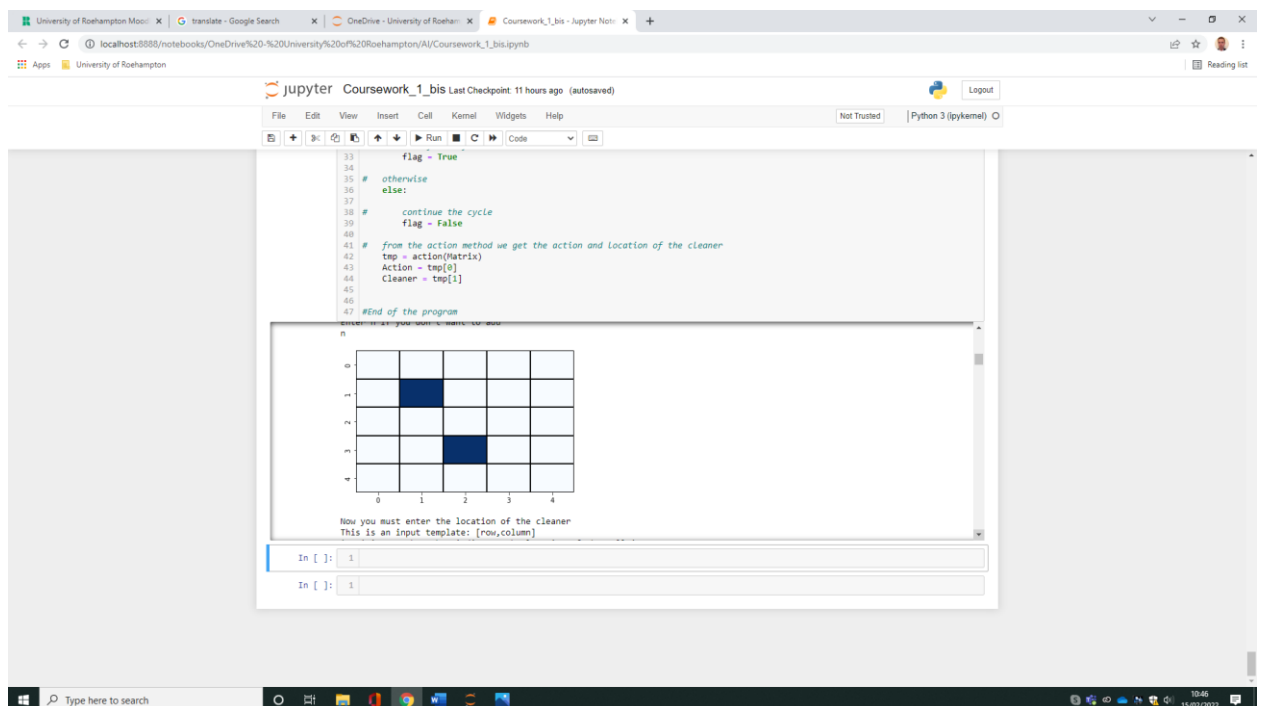
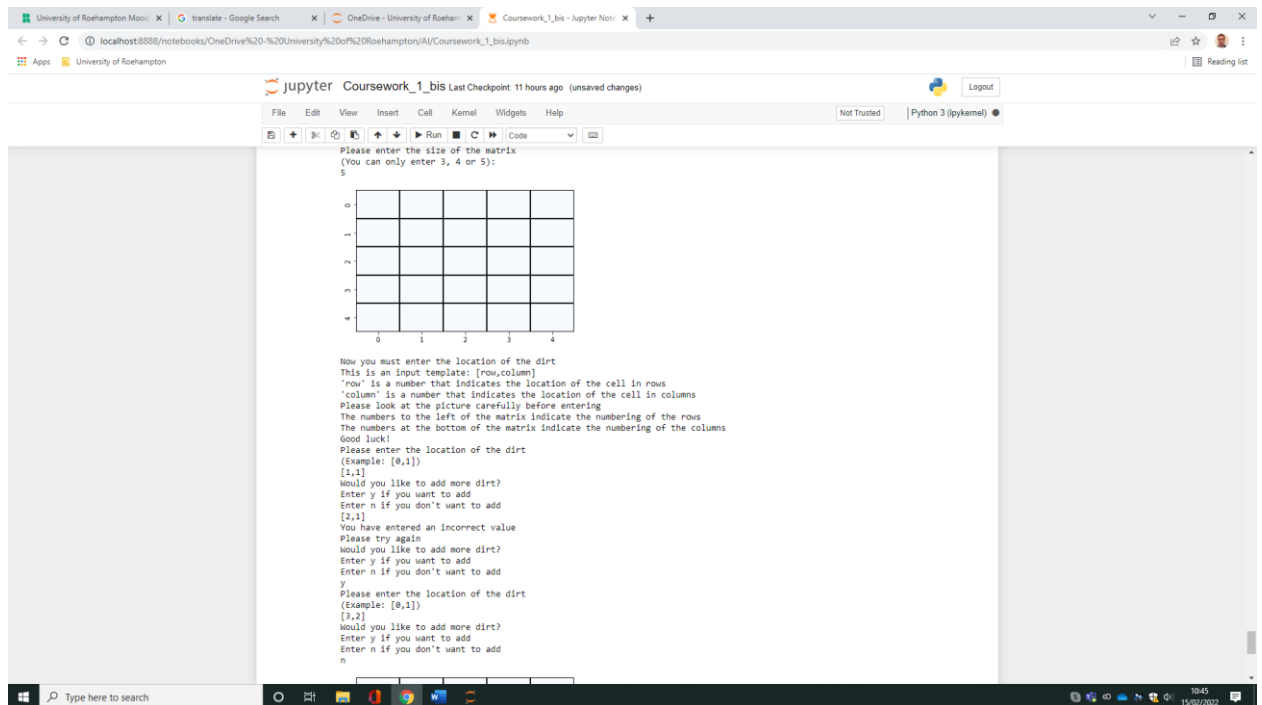


When all input from the user is in my program, the agent starts to move. Function "action(matrix)" start to find the agent coordinates in the matrix using function "action_location_cleaner(matrix)". Firstly it tries to find number 2 because this represents dirt and robot. After, it tries to find number 1 because it represents only robot. "action_location_cleaner(matrix)" can not have errors because it gets executed without human intervention, but in case I print to the user "Error! \nSomething wrong! \nRestart the program". When "location_cleaner" is found, "action(matrix)" invokes the function "action_new_action(matrix, location_cleaner)". This function represents the movement of the agent in the matrix. The agent can start to move left or right depending on the initial location of the agent. Picture by hand helps to understand how my agent moves. And this hand made scheme help me to write code (if statements with correct indentation). From "action_new_action(matrix, location_cleaner)" function, 3 means move right and 2 means move left.



Function responsible for cleaning is "do_action(matrix, action)". This function gives instructions about what to do the agent and where to do it. Moreover, this function changes the colour of the blues in the matrix after cleaning. And it uses the "check_matrix_to_dirt(matrix)" function to find dirt on the entire matrix, returning True or False (clean or not clean the matrix) that is used in the condition at the beginning of "do_action(...)". In the driving block (last one in the Jupiter notebook), "do_action(matrix, action)" is in the while loop until ALL the matrix is clean, then the while loop from the driving block finish and the program ends.

Code that is commented “`action_dirt_search(matrix, location_cleaner)`” is unfinished because my next step is to make the agent looks intelligent. My program moves agent cell by cell in the horizontal direction (left or right), but my program can be improved with the agent moving up and down before finishing the row because dirt is up or down in the neighbour row while moving horizontally the agent. This feature will be great to add, but I need more time to develop it. Pictures and screenshots of Solution3 follow:



University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```

33     flag = True
34
35     # otherwise
36     else:
37
38     # continue the cycle
39     flag = False
40
41     # from the action method we get the action and Location of the cleaner
42     tmp = action(Matrix)
43     Action = tmp[0]
44     Cleaner = tmp[1]
45
46
47 #End of the program

```

this is an input template: (row,column)
 'row' is a number that indicates the location of the cell in rows
 'column' is a number that indicates the location of the cell in columns
 Please look at the picture carefully before entering
 The numbers to the left of the matrix indicate the numbering of the rows
 The numbers at the bottom of the matrix indicate the numbering of the columns
 Good Luck!
 Please enter the location of the cleaner
 (Example: [0,1])
 [0,2]

0					
1					
2					

In []: 1

In []: 1

Type here to search

10:46 15/02/2022

University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```

33     flag = True
34
35     # otherwise
36     else:
37
38     # continue the cycle
39     flag = False
40
41     # from the action method we get the action and Location of the cleaner
42     tmp = action(Matrix)
43     Action = tmp[0]
44     Cleaner = tmp[1]
45
46
47 #End of the program

```

Now the cleaner does: move left

0					
1					
2					
3					
4					

[1, 2]
 Now the cleaner does: move left

In []: 1

In []: 1

Type here to search

10:47 15/02/2022

University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

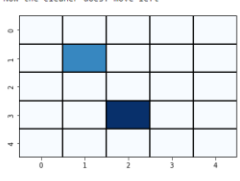
University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37
38 # continue the cycle
39 flag = False
40
41 # from the action method we get the action and Location of the cleaner
42 tmp = action(Matrix)
43 Action = tmp[0]
44 Cleaner = tmp[1]
45
46 #End of the program
47
```

Now the cleaner does: move left



[1, 1]
Now the cleaner does: clean

In []: 1

In []: 1

Type here to search

10:47 15/02/2022

University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

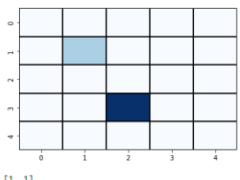
University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37
38 # continue the cycle
39 flag = False
40
41 # from the action method we get the action and Location of the cleaner
42 tmp = action(Matrix)
43 Action = tmp[0]
44 Cleaner = tmp[1]
45
46 #End of the program
47
```

[1, 1]
Now the cleaner does: clean



[1, 1]
Now the cleaner does: move left

In []: 1

In []: 1

Type here to search

10:47 15/02/2022

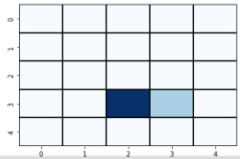
University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | localhost8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb | University of Roehampton | Reading list

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved) | Python 3 (ipykernel) | Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37
38 # continue the cycle
39 flag = False
40
41 # from the action method we get the action and location of the cleaner
42 tmp = action(Matrix)
43 Action = tmp[0]
44 Cleaner = tmp[1]
45
46 #End of the program
47
```

[3, 4]
Now the cleaner does: move left



In []: 1
In []: 1

Type here to search | 10:47 15/02/2022

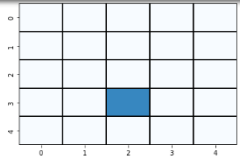
University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | localhost8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb | University of Roehampton | Reading list

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved) | Python 3 (ipykernel) | Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37
38 # continue the cycle
39 flag = False
40
41 # from the action method we get the action and location of the cleaner
42 tmp = action(Matrix)
43 Action = tmp[0]
44 Cleaner = tmp[1]
45
46 #End of the program
47
```

[3, 2]
Now the cleaner does: clean



In []: 1
In []: 1

Type here to search | 10:47 15/02/2022

University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

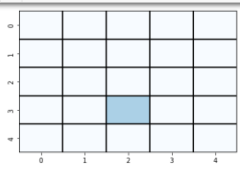
University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37     # continue the cycle
38     flag = False
39
40 # from the action method we get the action and location of the cleaner
41 tmp = action(Matrix)
42 Action = tmp[0]
43 Cleaner = tmp[1]
44
45
46
47 #End of the program
```



[3, 2]
The matrix is cleaned!

In []: 1

In []: 1

Type here to search

10:48 15/02/2022

University of Roehampton Moodle | translate - Google Search | OneDrive - University of Roehampton | Coursework_1_bis - Jupyter Notebook | +

localhost:8888/notebooks/OneDrive%20-%20University%20of%20Roehampton/AI/Coursework_1_bis.ipynb

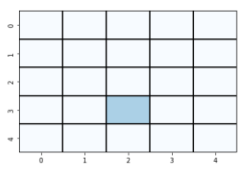
University of Roehampton

jupyter Coursework_1_bis Last Checkpoint: 11 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

```
33 flag = True
34
35 # otherwise
36 else:
37     # continue the cycle
38     flag = False
39
40 # from the action method we get the action and location of the cleaner
41 tmp = action(Matrix)
42 Action = tmp[0]
43 Cleaner = tmp[1]
44
45
46
47 #End of the program
```



[3, 2]
The matrix is cleaned!

In []: 1

In []: 1

Type here to search

10:48 15/02/2022

Regarding the questions from the task:

“o How did you decompose the problem and why?”

Small functions doing a specific task, except “do_action(matrix, action)” that do two tasks. Driving code is no function that uses all the previous designed functions.

“o Have you met any difficulties/bugs during this coursework, and how did you tackle it?”

A lot of difficulties because is long code and code jumps from one part to another. Google and books helped me a lot to design pieces of my program. There is no similar program on the Internet, but everything used is available on the Internet.

“o What did you think is the most difficult part of this coursework?”

Have the initial idea structure of the program. The code is simple, but implementing the logic is a little bit difficult, especially for disorganised people.

“o What have you learnt from investigating this problem?”

The “Divide and conquer” approach use many small functions. If loops are very powerful if right use and adequate indentation. Whiles loops same as ifs.

“o Which part(s) in your code design is your favourite and why?”

Designing the matrix because is nice graphically. And this function is only 3 lines of code. Solution1 and Solution2 were very easy to implement but is not vacuum cleaner code.

“o How did you improve the performance of your cleaner? If you had no time limit, how would you further improve your program?”

My agent is not optimized because it moves without checking dirt around it. And another obvious problem is that it jumps from the end of the matrix to the [0,0] cell in the matrix. This can not happen in real life.

“o Any references you might have used to complete the task.”

All references are included previously. Internet the most useful resource.

END

