

Dash Components



Objectives

After completing the lab you will be able to:

- Know how to add multiple graphs to the dashboard
- Work with Dash Callbacks to handle multiple outputs

Estimated time needed: 30 minutes

Dataset Used

[Airline Reporting Carrier On-Time Performance](#) dataset from [Data Asset eXchange](#)

About Skills Network Cloud IDE

This Skills Network Labs Cloud IDE (Integrated Development Environment) provides a hands-on environment in your web browser for completing course and project related labs. It utilizes Thelia, an open-source IDE platform, that can be run on desktop or on the cloud. So far in the course you have been using Jupyter notebooks to run your python code. This IDE provides an alternative for editing and running your Python code. In this lab you will be using this alternative Python runtime to create and launch your Dash applications.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. When you launch the Cloud IDE, you are presented with a 'dedicated computer on the cloud' exclusively for you. This is available to you as long as you are actively working on the labs.

Once you close your session or it is timed out due to inactivity, you are logged off, and this 'dedicated computer on the cloud' is deleted along with any files you may have created, downloaded or installed. The next time you launch this lab, a new environment is created for you.

If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan to your time accordingly and finish your labs in a single session.

Let's start creating dash application

Theme

Analyze flight delays in a dashboard.

Dashboard Components

- Monthly average carrier delay by reporting airline for the given year.
- Monthly average weather delay by reporting airline for the given year.
- Monthly average national air system delay by reporting airline for the given year.
- Monthly average security delay by reporting airline for the given year.
- Monthly average late aircraft delay by reporting airline for the given year.

NOTE: Year range should be between 2010 and 2020

Expected Output

Below is the expected result from the lab. Our dashboard application consists of three components:

- Title of the application
- Component to enter input year
- 5 Charts conveying the different types of flight delay. Chart section is divided into three segments.
 - Carrier and Weather delay in the first segment
 - National air system and Security delay in the second segment
 - Late aircraft delay in the third segment



To do:

- Design layout for the application.
- Create a callback function. Add callback decorator, define inputs and outputs.
- Review the helper function that performs computation on the provided inputs.
- Create 5 line graphs.
- Run the application.

Get the tool ready

- Install python packages required to run the application. Copy and paste the below command to the terminal.

```
1. python3.8 -m pip install packaging
```


```
1. pip install https://0.20 dash plotly

Copied

root@headocir-makias:~/home/project_X
```

```
theig@headocir-makias:~/home/project_X$ pip install https://0.20 dash plotly
/usr/lib/python3/dist-packages/setuptools/dist.py:319: CryptographyDeprecationWarning: int from bytes is deprecated, use int.from_bytes instead
/usr/lib/python3/dist-packages/setuptools/dist.py:319: CryptographyDeprecationWarning: int from bytes is deprecated, use int.from_bytes instead
/usr/lib/python3/dist-packages/setuptools/dist.py:319: CryptographyDeprecationWarning: int from bytes is deprecated, use int.from_bytes instead
/usr/lib/python3/dist-packages/setuptools/dist.py:319: CryptographyDeprecationWarning: int from bytes is deprecated, use int.from_bytes instead
defaulting to user installation because normal site-packages is not writable
Collecting https://0.20
  Downloading https://0.20.py3-none-any.whl (42 kB)
    82 kb 779 k/s
Collecting dash
  Downloading dash-2.0.1-py3-none-any.whl (9.9 MB)
    9.9 MB 40.7 MB/s
Collecting plotly
  Downloading plotly-5.10.0-py3-py3-none-any.whl (15.2 MB)
    15.2 MB 39.3 MB/s
Requirement already satisfied: sniffio in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (2.1.0)
Requirement already satisfied: httpcore<0.14.0,>=0.13.1 in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (0.11.2)
Requirement already satisfied: asynio-generator in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (1.10)
Requirement already satisfied: certifi in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (2020.12.5)
Requirement already satisfied: f3r3ss0rd[all20208]@2.2.1.1 in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (1.5.0)
Requirement already satisfied: cursive-normalizer in /home/theia/.local/lib/python3.6/site-packages (from https://0.20) (2.0.12)
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
```

-
- A screenshot of the Visual Studio Code interface, specifically the Explorer sidebar on the left. The sidebar has a dark background. At the top, there's a header with 'EXPLORER' and a search icon. Below it, there's a section labeled 'OPEN EDITORS' with a plus icon. The main area of the sidebar is labeled 'PROJECT' and contains a tree view of files and folders. A yellow box highlights the Explorer icon in the Activity Bar on the far left. A yellow circle with the number '1' is placed over the Explorer icon in the Explorer sidebar. A yellow circle with the number '2' is placed over the Explorer icon in the Explorer sidebar. A yellow arrow points from the Explorer icon in the Explorer sidebar to the Explorer icon in the Explorer sidebar.

- 

-
- The screenshot shows the PyCharm application window. The 'Terminal' menu is open, displaying a list of options. The 'Run' option is highlighted, which has opened a submenu. The submenu contains the following items: 'Run Task...', 'Run Build Task...', 'Run Test Task...', 'Run Last Task...', 'Show Running Tasks...', 'Restart Running Task...', 'Terminate Task...', and 'Attach Task...'. The 'Run' option in the main menu has a keyboard shortcut of '⇧ R' (Shift+R). The 'Run' option in the submenu is highlighted in blue.

-
- The screenshot shows a code editor window with a menu bar at the top containing 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. On the left side, there is a vertical toolbar with icons for file operations. The main editing area has a title bar that reads 'flight_delay.py x'. Inside the editor, there is a single line of code, the number '1'.

- Importing necessary libraries
- Reading the data

1. 1
2. 2

```
1. 3
2. 4
3. 5
4. 6
5. 7
6. 8
7. 9
8. 10
9. 11
10. 12
11. 13
12. 14
13. 15
14. 16
15. 17
16. 18
17. 19
18. 20
19. 21
20. 22
21. 23
22. 24
23. 25
24. 26
25. 27
26. 28
27. 29
28. 30
29. 31
30. 32
31. 33
32. 34
33. 35
34. 36
35. 37
36. 38
37. 39
38. 40
39. 41
40. 42
41. 43
42. 44
43. 45
44. 46
45. 47
46. 48
47. 49
48. 50
49. 51
50. 52
51. 53
52. 54
53. 55
54. 56
55. 57
56. 58
57. 59
58. 60
59. 61
60. 62
61. 63
62. 64
63. 65
64. 66
65. 67
66. 68
67. 69
68. 70
69. 71
70. 72
71. 73
72. 74
73. 75
74. 76
75. 77
76. 78
77. 79
78. 80
79. 81
80. 82
81. 83
82. 84
83. 85
84. 86
85. 87
86. 88
87. 89
88. 90
89. 91
90. 92
91. 93
92. 94
93. 95
94. 96
95. 97
96. 98
97. 99
98. 100
99. 101
100. 102
101. 103
102. 104
103. 105
104. 106
105. 107
106. 108
107. 109
108. 110
109. 111
110. 112
111. 113
112. 114
113. 115
114. 116
115. 117
116. 118
117. 119
118. 120
119. 121
120. 122
121. 123
122. 124
123. 125
124. 126
125. 127
126. 128
127. 129
128. 130
129. 131
130. 132
131. 133
132. 134
133. 135
134. 136
135. 137
136. 138
137. 139
138. 140
139. 141
140. 142
141. 143
142. 144
143. 145
144. 146
145. 147
146. 148
147. 149
148. 150
149. 151
150. 152
151. 153
152. 154
153. 155
154. 156
155. 157
156. 158
157. 159
158. 160
159. 161
160. 162
161. 163
162. 164
163. 165
164. 166
165. 167
166. 168
167. 169
168. 170
169. 171
170. 172
171. 173
172. 174
173. 175
174. 176
175. 177
176. 178
177. 179
178. 180
179. 181
180. 182
181. 183
182. 184
183. 185
184. 186
185. 187
186. 188
187. 189
188. 190
189. 191
190. 192
191. 193
192. 194
193. 195
194. 196
195. 197
196. 198
197. 199
198. 200
199. 201
200. 202
201. 203
202. 204
203. 205
204. 206
205. 207
206. 208
207. 209
208. 210
209. 211
210. 212
211. 213
212. 214
213. 215
214. 216
215. 217
216. 218
217. 219
218. 220
219. 221
220. 222
221. 223
222. 224
223. 225
224. 226
225. 227
226. 228
227. 229
228. 230
229. 231
230. 232
231. 233
232. 234
233. 235
234. 236
235. 237
236. 238
237. 239
238. 240
239. 241
240. 242
241. 243
242. 244
243. 245
244. 246
245. 247
246. 248
247. 249
248. 250
249. 251
250. 252
251. 253
252. 254
253. 255
254. 256
255. 257
256. 258
257. 259
258. 260
259. 261
260. 262
261. 263
262. 264
263. 265
264. 266
265. 267
266. 268
267. 269
268. 270
269. 271
270. 272
271. 273
272. 274
273. 275
274. 276
275. 277
276. 278
277. 279
278. 280
279. 281
280. 282
281. 283
282. 284
283. 285
284. 286
285. 287
286. 288
287. 289
288. 290
289. 291
290. 292
291. 293
292. 294
293. 295
294. 296
295. 297
296. 298
297. 299
298. 300
299. 301
300. 302
301. 303
302. 304
303. 305
304. 306
305. 307
306. 308
307. 309
308. 310
309. 311
310. 312
311. 313
312. 314
313. 315
314. 316
315. 317
316. 318
317. 319
318. 320
319. 321
320. 322
321. 323
322. 324
323. 325
324. 326
325. 327
326. 328
327. 329
328. 330
329. 331
330. 332
331. 333
332. 334
333. 335
334. 336
335. 337
336. 338
337. 339
338. 340
339. 341
340. 342
341. 343
342. 344
343. 345
344. 346
345. 347
346. 348
347. 349
348. 350
349. 351
350. 352
351. 353
352. 354
353. 355
354. 356
355. 357
356. 358
357. 359
358. 360
359. 361
360. 362
361. 363
362. 364
363. 365
364. 366
365. 367
366. 368
367. 369
368. 370
369. 371
370. 372
371. 373
372. 374
373. 375
374. 376
375. 377
376. 378
377. 379
378. 380
379. 381
380. 382
381. 383
382. 384
383. 385
384. 386
385. 387
386. 388
387. 389
388. 390
389. 391
390. 392
391. 393
392. 394
393. 395
394. 396
395. 397
396. 398
397. 399
398. 400
399. 401
400. 402
401. 403
402. 404
403. 405
404. 406
405. 407
406. 408
407. 409
408. 410
409. 411
410. 412
411. 413
412. 414
413. 415
414. 416
415. 417
416. 418
417. 419
418. 420
419. 421
420. 422
421. 423
422. 424
423. 425
424. 426
425. 427
426. 428
427. 429
428. 430
429. 431
430. 432
431. 433
432. 434
433. 435
434. 436
435. 437
436. 438
437. 439
438. 440
439. 441
440. 442
441. 443
442. 444
443. 445
444. 446
445. 447
446. 448
447. 449
448. 450
449. 451
450. 452
451. 453
452. 454
453. 455
454. 456
455. 457
456. 458
457. 459
458. 460
459. 461
460. 462
461. 463
462. 464
463. 465
464. 466
465. 467
466. 468
467. 469
468. 470
469. 471
470. 472
471. 473
472. 474
473. 475
474. 476
475. 477
476. 478
477. 479
478. 480
479. 481
480. 482
481. 483
482. 484
483. 485
484. 486
485. 487
486. 488
487. 489
488. 490
489. 491
490. 492
491. 493
492. 494
493. 495
494. 496
495. 497
496. 498
497. 499
498. 500
499. 501
500. 502
501. 503
502. 504
503. 505
504. 506
505. 507
506. 508
507. 509
508. 510
509. 511
510. 512
511. 513
512. 514
513. 515
514. 516
515. 517
516. 518
517. 519
518. 520
519. 521
520. 522
521. 523
522. 524
523. 525
524. 526
525. 527
526. 528
527. 529
528. 530
529. 531
530. 532
531. 533
532. 534
533. 535
534. 536
535. 537
536. 538
537. 539
538. 540
539. 541
540. 542
541. 543
542. 544
543. 545
544. 546
545. 547
546. 548
547. 549
548. 550
549. 551
550. 552
551. 553
552. 554
553. 555
554. 556
555. 557
556. 558
557. 559
558. 560
559. 561
560. 562
561. 563
562. 564
563. 565
564. 566
565. 567
566. 568
567. 569
568. 570
569. 571
570. 572
571. 573
572. 574
573. 575
574. 576
575. 577
576. 578
577. 579
578. 580
579. 581
580. 582
581. 583
582. 584
583. 585
584. 586
585. 587
586. 588
587. 589
588. 590
589. 591
590. 592
591. 593
592. 594
593. 595
594. 596
595. 597
596. 598
597. 599
598. 600
599. 601
600. 602
601. 603
602. 604
603. 605
604. 606
605. 607
606. 608
607. 609
608. 610
609. 611
610. 612
611. 613
612. 614
613. 615
614. 616
615. 617
616. 618
617. 619
618. 620
619. 621
620. 622
621. 623
622. 624
623. 625
624. 626
625. 627
626. 628
627. 629
628. 630
629. 631
630. 632
631. 633
632. 634
633. 635
634. 636
635. 637
636. 638
637. 639
638. 640
639. 641
640. 642
641. 643
642. 644
643. 645
644. 646
645. 647
646. 648
647. 649
648. 650
649. 651
650. 652
651. 653
652. 654
653. 655
654. 656
655. 657
656. 658
657. 659
658. 660
659. 661
660. 662
661. 663
662. 664
663. 665
664. 666
665. 667
666. 668
667. 669
668. 670
669. 671
670. 672
671. 673
672. 674
673. 675
674. 676
675. 677
676. 678
677. 679
678. 680
679. 681
680. 682
681. 683
682. 684
683. 685
684. 686
685. 687
686. 688
687. 689
688. 690
689. 691
690. 692
691. 693
692. 694
693. 695
694. 696
695. 697
696. 698
697. 699
698. 700
699. 701
700. 702
701. 703
702. 704
703. 705
704. 706
705. 707
706. 708
707. 709
708. 710
709. 711
710. 712
711. 713
712. 714
713. 715
714. 716
715. 717
716. 718
717. 719
718. 720
719. 721
720. 722
721. 723
722. 724
723. 725
724. 726
725. 727
726. 728
727. 729
728. 730
729. 731
730. 732
731. 733
732. 734
733. 735
734. 736
735. 737
736. 738
737. 739
738. 740
739. 741
740. 742
741. 743
742. 744
743. 745
744. 746
745. 747
746. 748
747. 749
748. 750
749. 751
750. 752
751. 753
752. 754
753. 755
754. 756
755. 757
756. 758
757. 759
758. 760
759. 761
760. 762
761. 763
762. 764
763. 765
764. 766
765. 767
766. 768
767. 769
768. 770
769. 771
770. 772
771. 773
772. 774
773. 775
774. 776
775. 777
776. 778
777. 779
778. 780
779. 781
780. 782
781. 783
782. 784
783. 785
784. 786
785. 787
786. 788
787. 789
788. 790
789. 791
790. 792
791. 793
792. 794
793. 795
794. 796
795. 797
796. 798
797. 799
798. 800
799. 801
800. 802
801. 803
802. 804
803. 805
804. 806
805. 807
806. 808
807. 809
808. 810
809. 811
810. 812
811. 813
812. 814
813. 815
814. 816
815. 817
816. 818
817. 819
818. 820
819. 821
820. 822
821. 823
822. 824
823. 825
824. 826
825. 827
826. 828
827. 829
828. 830
829. 831
830. 832
831. 833
832. 834
833. 835
834. 836
835. 837
836. 838
837. 839
838. 840
839. 841
840. 842
841. 843
842. 844
843. 845
844. 846
845. 847
846. 848
847. 849
848. 850
849. 851
850. 852
851. 853
852. 854
853. 855
854. 856
855. 857
856. 858
857. 859
858. 860
859. 861
860. 862
861. 863
862. 864
863. 865
864. 866
865. 867
866. 868
867. 869
868. 870
869. 871
870. 872
871. 873
872. 874
873. 875
874. 876
875. 877
876. 878
877. 879
878. 880
879. 881
880. 882
881. 883
882. 884
883. 885
884. 886
885. 887
886. 888
887. 889
888. 890
889. 891
890. 892
891. 893
892. 894
893. 895
894. 896
895. 897
896. 898
897. 899
898. 900
899. 901
900. 902
901. 903
902. 904
903. 905
904. 906
905. 907
906. 908
907. 909
908. 910
909. 911
910. 912
911. 913
912. 914
913. 915
914. 916
915. 917
916. 918
917. 919
918. 920
919. 921
920. 922
921. 923
922. 924
923. 925
924. 926
925. 927
926. 928
927. 929
928. 930
929. 931
930. 932
931. 933
932. 934
933. 935
934. 936
935. 937
936. 938
937. 939
938. 940
939. 941
940. 942
941. 943
942. 944
943. 945
944. 946
945. 947
946. 948
947. 949
948. 950
949. 951
950. 952
951. 953
952. 954
953. 955
954. 956
955. 957
956. 958
957. 959
958. 960
959. 961
960. 962
961. 963
962. 964
963. 965
964. 966
965. 967
966. 968
967. 969
968. 970
969. 971
970. 972
971. 973
972. 974
973. 975
974. 976
975. 977
976. 978
977. 979
978. 980
979. 981
980. 982
981. 983
982. 984
983. 985
984. 986
985. 987
986. 988
987. 989
988. 990
989. 991
990. 992
991. 993
992. 994
993. 995
994. 996
995. 997
996. 998
997. 999
998. 1000
999. 1001
1000. 1002
```

Copied

TASK 2 - Create dash application and get the layout skeleton

Next, we create a skeleton for our dash application. Our dashboard application layout has three components as seen before:

- Title of the application
- Component to enter input year inside a layout division
- 5 Charts conveying the different types of flight delay

Mapping to the respective Dash HTML tags:

- Title added using `html.H1()` tag
- Layout division added using `html.Div()` and input component added using `dcc.Input()` tag inside the layout division.
- 5 charts split into three segments. Each segment has a layout division added using `html.Div()` and chart added using `dcc.Graph()` tag inside the layout division.

Copy the below code to the `flight_delay.py` script and review the structure.

NOTE: Copy below the current code

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. 53
54. 54
55. 55
56. 56
57. 57
58. 58
59. 59
60. 60
61. 61
62. 62
63. 63
64. 64
65. 65
66. 66
67. 67
68. 68
69. 69
70. 70
71. 71
72. 72
73. 73
74. 74
75. 75
76. 76
77. 77
78. 78
79. 79
80. 80
81. 81
82. 82
83. 83
84. 84
85. 85
86. 86
87. 87
88. 88
89. 89
90. 90
91. 91
92. 92
93. 93
94. 94
95. 95
96. 96
97. 97
98. 98
99. 99
100. 100
101. 101
102. 102
103. 103
104. 104
105. 105
106. 106
107. 107
108. 108
109. 109
110. 110
111. 111
112. 112
113. 113
114. 114
115. 115
116. 116
117. 117
118. 118
119. 119
120. 120
121. 121
122. 122
123. 123
124. 124
125. 125
126. 126
127. 127
128. 128
129. 129
130. 130
131. 131
132. 132
133. 133
134. 134
135. 135
136. 136
137. 137
138. 138
139. 139
140. 140
141. 141
142. 142
143. 143
144. 144
145. 145
146. 146
147. 147
148. 148
149. 149
150. 150
151. 151
152. 152
153. 153
154. 154
155. 155
156. 156
157. 157
158. 158
159. 159
160. 160
161. 161
162. 162
163. 163
164. 164
165. 165
166. 166
167. 167
168. 168
169. 169
170. 170
171. 171
172. 172
173. 173
174. 174
175. 175
176. 176
177. 177
178. 178
179. 179
180. 180
181. 181
182. 182
183. 183
184. 184
185. 185
186. 186
187. 187
188. 188
189. 189
190. 190
191. 191
192. 192
193. 193
194. 194
195. 195
196. 196
197. 197
198. 198
199. 199
200. 200
201. 201
202. 202
203. 203
204. 204
205. 205
206. 206
207. 207
208. 208
209. 209
210. 210
211. 211
212. 212
213. 213
214. 214
215. 215
216. 216
217. 217
218. 218
219. 219
220. 220
221. 221
222. 222
223. 223
224. 224
225. 225
226. 226
227. 227
228. 228
229. 229
230. 230
231. 231
232. 232
233. 233
234. 234
235. 235
236. 236
237. 237
238. 238
239. 239
240. 240
241. 241
242. 242
243. 243
244. 244
245. 245
246. 246
247. 247
248. 248
249. 249
250. 250
251. 251
252. 252
253. 253
254. 254
255. 255
256. 256
257. 257
258. 258
259. 259
260. 260
261. 261
262. 262
263. 263
264. 264
265. 265
266. 266
267. 267
268. 268
269. 269
270. 270
271. 271
272. 272
273. 273
274. 274
275. 275
276. 276
277. 277
278. 278
279. 279
280. 280
281. 281
282. 282
283. 283
284. 284
285. 285
286. 286
287. 287
288. 288
289. 289
290. 290
291. 291
292. 292
293. 293
294. 294
295. 295
296. 296
297. 297
298. 298
299. 299
300. 300
301. 301
302. 302
303. 303
304. 304
305. 305
306. 306
307. 307
308. 308
309. 309
310. 310
311. 311
312. 312
313. 313
314. 314
315. 315
316. 316
317. 317
318. 318
319. 319
320. 320
321. 321
322. 322
323. 323
324. 324
325. 325
326. 326
327. 327
328. 328
329. 329
330. 330
331. 331
332. 332
333. 333
334. 334
335. 335
336. 336
337. 337
338. 338
339. 339
340. 340
341. 341
342. 342
343. 343
344. 344
345. 345
346. 346
347. 347
348. 348
349. 349
350. 350
351. 351
352. 352
353. 353
354. 354
3
```

```
19: 19
20: 20
21: 21
22: 22
23:
24: 1. """ Compute_info function description
25: 2. This function takes in airline data and selected year as an input and performs computation for creating charts and plots.
26: 3.
27: 4. Arguments:
28: 5.     airline_data: Input airline data.
29: 6.     entered_year: Input year for which computation needs to be performed.
30: 7.
31: 8. Returns:
32: 9.     Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.
33: 10.
34: 11. """
35: 12. def compute_info(airline_data, entered_year):
36: 13.     # Select data
37: 14.     df = airline_data[airline_data['year']==int(entered_year)]
38: 15.     # Compute delay averages
39: 16.     avg_car = df.groupby(['Month', 'Reporting Airline'])['CarrierDelay'].mean().reset_index()
40: 17.     avg_weather = df.groupby(['Month', 'Reporting Airline'])['WeatherDelay'].mean().reset_index()
41: 18.     avg_NAS = df.groupby(['Month', 'Reporting Airline'])['NASDelay'].mean().reset_index()
42: 19.     avg_sec = df.groupby(['Month', 'Reporting Airline'])['SecurityDelay'].mean().reset_index()
43: 20.     avg_late = df.groupby(['Month', 'Reporting Airline'])['LateAircraftDelay'].mean().reset_index()
44: 21.     return avg_car, avg_weather, avg_NAS, avg_sec, avg_late
45: 22.
```

Copied

TASK 5 - Add the application callback function

The core idea of this application is to get year as user input and update the dashboard in real-time. We will be using callback function for the same.

Steps:

- Define the callback decorator
- Define the callback function that uses the input provided to perform the computation
- Create graph and return it as an output
- Run the application

Copy the below code to the flight_delay.py script and review the structure.

NOTE: Copy below the current code

```
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
10: 10
11: 11
12: 12
13: 13
14: 14
15: 15
16: 16
17: 17
18: 18
19: 19
20: 20
21: 21
22: 22
23: 23
24: 24
25: 25
26: 26
27: 27
28: 28
29: 29
30: 30
31: 31
32:
33: 1. # Callback decorator
34: 2. @app.callback([
35: 3.     Output(component_id='carrier-plot', component_property='figure'),
36: 4.     ...
37: 5.     ...
38: 6.     ...
39: 7.     ...
40: 8. ],
41: 9.     Input(...))
42: 10. # Computation to callback function and return graph
43: 11. def get_graph(entered_year):
44: 12.
45: 13.     # Compute required information for creating graph from the data
46: 14.     avg_car, avg_weather, avg_NAS, avg_sec, avg_late = compute_info(airline_data, entered_year)
47: 15.
48: 16.     # Line plot for carrier delay
49: 17.     carrier_fig = px.line(avg_car, x='Month', y='CarrierDelay', color='Reporting Airline', title='Average carrier delay time (minutes) by airline')
50: 18.     # Line plot for weather delay
51: 19.     weather_fig = px.line(avg_weather, x='Month', y='WeatherDelay', color='Reporting Airline', title='Average weather delay time (minutes) by airline')
52: 20.     # Line plot for NAS delay
53: 21.     nas_fig = px.line(avg_NAS, x='Month', y='NASDelay', color='Reporting Airline', title='Average NAS delay time (minutes) by airline')
54: 22.     # Line plot for security delay
55: 23.     sec_fig = px.line(avg_sec, x='Month', y='SecurityDelay', color='Reporting Airline', title='Average security delay time (minutes) by airline')
56: 24.     # Line plot for late aircraft delay
57: 25.     late_fig = px.line(avg_late, x='Month', y='LateAircraftDelay', color='Reporting Airline', title='Average late aircraft delay time (minutes) by airline')
58: 26.
59: 27.     return(carrier_fig, weather_fig, nas_fig, sec_fig, late_fig)
60: 28.
61: 29. # Run the app
62: 30. if __name__ == '__main__':
63: 31.     app.run_server()
```

Copied

TASK 6 - Update the callback function

Callback decorator

- Refer examples provided [here](#)
- We have 5 output component added in a list. Update output component id parameter with the ids provided in the dcc.graph() component and set the component property as figure. One sample has been added to the skeleton.
- Update input component id parameter with the id provided in the dcc.input() component and component property as value.

Callback function

Next is to update the get_graph function. We have already added a function compute_info that will perform computation on the data using the input.

Mapping the returned value from the function compute_info to graph:

- avg_car - input for carrier delay
- avg_weather - input for weather delay
- avg_NAS - input for NAS delay
- avg_sec - input for security delay
- avg_late - input for late aircraft delay

Code has been provided for plotting carrier delay. Follow the same process and use the above mapping to get plots for other 4 delays.

Refer to the full code of 4_Flight_Delay_Time_Statistics_Dashboard.py

```
1: 1
2: 2
3: 3
4: 4
5: 5
6: 6
7: 7
8: 8
9: 9
10: 10
11: 11
12: 12
13: 13
14: 14
15: 15
16: 16
17: 17
18: 18
19: 19
20: 20
21: 21
22: 22
23: 23
24: 24
25: 25
26: 26
27: 27
28: 28
29: 29
30: 30
31: 31
32: 32
33: 33
34: 34
35: 35
36: 36
37: 37
38: 38
39: 39
40: 40
41: 41
42: 42
43: 43
44: 44
45: 45
46: 46
47: 47
48: 48
49: 49
50: 50
51: 51
52: 52
53: 53
54: 54
55: 55
56: 56
57: 57
58: 58
59: 59
60: 60
61: 61
62: 62
63: 63
64: 64
65: 65
66: 66
67: 67
68: 68
69: 69
70: 70
71: 71
72: 72
73: 73
74: 74
75: 75
76: 76
77: 77
78: 78
79: 79
80: 80
81: 81
82: 82
83: 83
```

```

84. 84
85. 85
86. 86
87. 87
88. 88
89. 89
90. 90
91. 91
92. 92
93. 93
94. 94
95. 95
96. 96
97. 97
98. 98
99. 99
100. 100
101. 101
102. 102
103. 103
104. 104
105. 105
106. 106

1. # Import required libraries
2. import pandas as pd
3. import dash
4. from dash import dcc
5. from dash import html
6. from dash.dependencies import Input, Output
7. import plotly.express as px
8.
9. # Read the airline data into pandas dataframe
10. airline_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillNetwork-Dataset2of16/airline_data.csv',
11.                             encoding='UTF-8')
12.
13. # Create a dash application
14. app = dash.Dash(__name__)
15.
16. # Build dash app layout
17.
18. app.layout = html.Div(children=[html.H1('Flight Delay Time Statistics',
19.                                         style={'text-align': 'center', 'color': '#000036',
20.                                              'font-size': 30}),
21.                               html.Div([
22.                                   dcc.Input(id='input-year', value='2018',
23.                                             type='text', style={'height': '35px', 'font-size': 30}),
24.                                   html.Div([
25.                                       html.Div([
26.                                           # Segment 1
27.                                           html.Div([
28.                                               html.Div(dcc.Graph(id='carrier-plot')),
29.                                               html.Div(dcc.Graph(id='weather-plot'))
30.                                           ], style={'display': 'flex'}),
31.                                           # Segment 2
32.                                           html.Div([
33.                                               html.Div(dcc.Graph(id='nas-plot')),
34.                                               html.Div(dcc.Graph(id='security-plot'))
35.                                           ], style={'display': 'flex'}),
36.                                           # Segment 3
37.                                           html.Div(dcc.Graph(id='late-plot'), style={'width': '60%'})
38.                                       ]),
39.                                       # Segment 4
40.                                       html.Div(dcc.Graph(id='late-plot'), style={'width': '60%'})
41.                                   ]),
42.                               ])
43.
44. *** Compute_info function description
45.
46. This function takes in airline data and selected year as an input and performs computation for creating charts and plots.
47.
48. Arguments:
49.     airline_data: Input airline data.
50.     entered_year: Input year for which computation needs to be performed.
51.
52. Returns:
53.     Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.
54.
55. ***
56. def compute_info(airline_data, entered_year):
57.     # Select data
58.     df = airline_data[airline_data['Year']==int(entered_year)]
59.
60.     # Compute delay averages
61.     avg_car = df.groupby(['Month', 'Reporting_Airline'])['CarrierDelay'].mean().reset_index()
62.     avg_weather = df.groupby(['Month', 'Reporting_Airline'])['WeatherDelay'].mean().reset_index()
63.     avg_nas = df.groupby(['Month', 'Reporting_Airline'])['NASDelay'].mean().reset_index()
64.     avg_sec = df.groupby(['Month', 'Reporting_Airline'])['SecurityDelay'].mean().reset_index()
65.     avg_late = df.groupby(['Month', 'Reporting_Airline'])['LateAircraftDelay'].mean().reset_index()
66.
67.     return avg_car, avg_weather, avg_nas, avg_sec, avg_late
68.
69. ***Callback Function
70.
71. Function that returns figures using the provided input year.
72.
73. Arguments:
74.     entered_year: Input year provided by the user.
75.
76. Returns:
77.     List of figures computed using the provided helper function 'compute_info'.
78.
79. ***
80. @app.callback(
81.     Output(component_id='carrier-plot', component_property='figure'),
82.     Output(component_id='weather-plot', component_property='figure'),
83.     Output(component_id='nas-plot', component_property='figure'),
84.     Output(component_id='security-plot', component_property='figure'),
85.     Output(component_id='late-plot', component_property='figure'),
86.     Input(component_id='input-year', component_property='value')
87. )
88.
89. # Computation to callback function and return graph
90. def get_graph(entered_year):
91.
92.     # Compute required information for creating graph from the data
93.     avg_car, avg_weather, avg_nas, avg_sec, avg_late = compute_info(airline_data, entered_year)
94.
95.     # Line plot for carrier delay
96.     carrier_fig = px.line(avg_car, x='Month', y='CarrierDelay', color='Reporting_Airline', title='Average carrier delay time (minutes) by airline')
97.
98.     # Line plot for weather delay
99.     weather_fig = px.line(avg_weather, x='Month', y='WeatherDelay', color='Reporting_Airline', title='Average weather delay time (minutes) by airline')
100.
101.     # Line plot for nas delay
102.     nas_fig = px.line(avg_nas, x='Month', y='NASDelay', color='Reporting_Airline', title='Average NAS delay time (minutes) by airline')
103.
104.     # Line plot for security delay
105.     sec_fig = px.line(avg_sec, x='Month', y='SecurityDelay', color='Reporting_Airline', title='Average security delay time (minutes) by airline')
106.
107.     # Line plot for late aircraft delay
108.     late_fig = px.line(avg_late, x='Month', y='LateAircraftDelay', color='Reporting_Airline', title='Average late aircraft delay time (minutes) by airline')
109.
110.     return [carrier_fig, weather_fig, nas_fig, sec_fig, late_fig]
111.
112.
113.
114. # Run the app
115. if __name__ == '__main__':
116.     app.run_server()

```

[Copy](#)

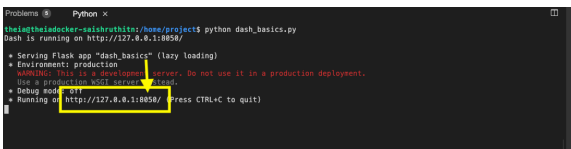
TASK 6 - Run the application

- Copy and paste the below command in the terminal to run the application.

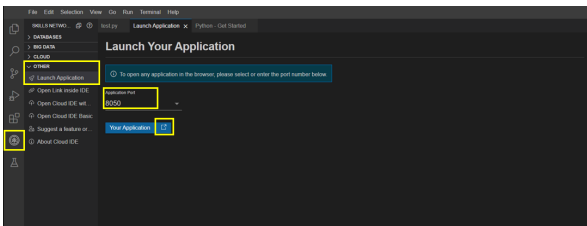
```
1. python3.8 Flight_delay.py
```

[Copy](#)

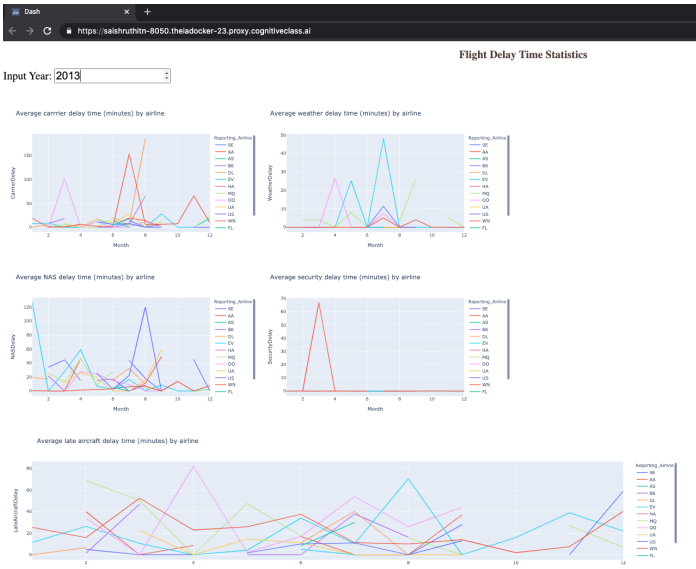
- Observe the port number shown in the terminal.



- Click on the Launch Application option from the side menu bar. Provide the port number and click ok



The app will open in a new browser tab like below:



Congratulations, you have successfully created your dash application!

Exercise : Practice Tasks

You will practice some tasks to update the dashboard.

1. Change the title to the dashboard from "Flight Delay Time Statistics" to "Flight Details Statistics Dashboard" using HTML H1 component and font-size as 35.
- Answer
2. Save the above changes and rename file as #flight_details.py and relaunch the dashboard application to see the updated dashboard title.
- Answer
3. Write a command to stop the running app in the terminal
- Answer

Author

Saishruthi Swaminathan

© IBM Corporation. All rights reserved.