

This task1:1 was run in jupyter ; Other remaining tasks were run in PyCharm

New Tab | Home Page - Select or create a notebook | Untitled - Jupyter Notebook

localhost:8888/notebooks/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

```
In [5]: try:
        user_input = input("enter a string")
        if not user_input:
            raise ValueError('empty string!!!! Hence using default string!! Author: Harish \n')
        except ValueError as err:
            print(err)
            user_input = "Result \n 1.Install Jupyter notebook and run the first program and share the screenshot of the output."

        print(user_input)
```

enter a string
empty string!!!! Hence using default string!! Author: Harish

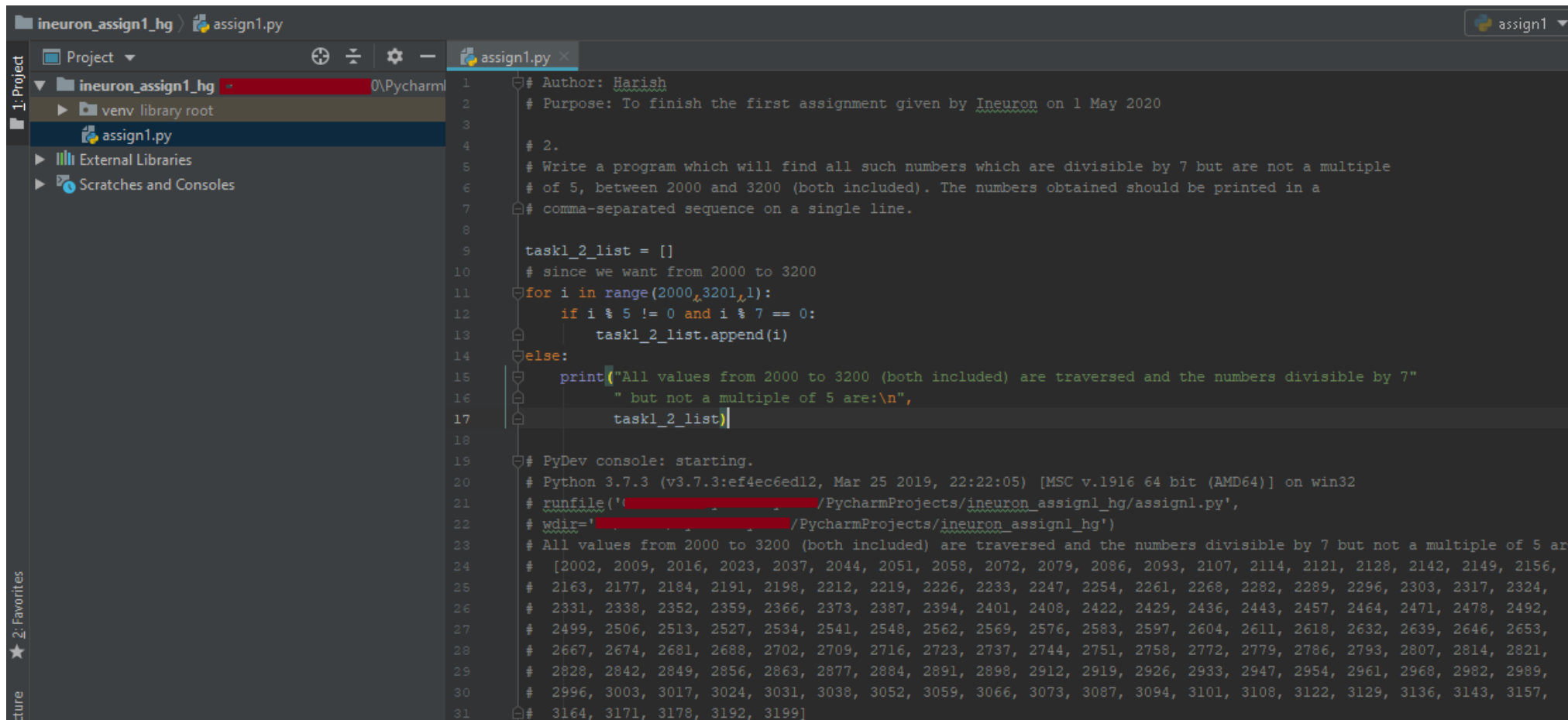
Result
1.Install Jupyter notebook and run the first program and share the screenshot of the output.

In []:

Have used two versions to solve the same . One of the answers doesn't have the function taught in the session , so wasn't sure if that would be acceptable for this , so have pasted both approaches

Approach 1: representing output as a list;

Please note, all the values are printed in a single line but since so many values couldn't be showed in this snapshot, have copied the results in separate lines with comments from the pycharm output console to the code editor



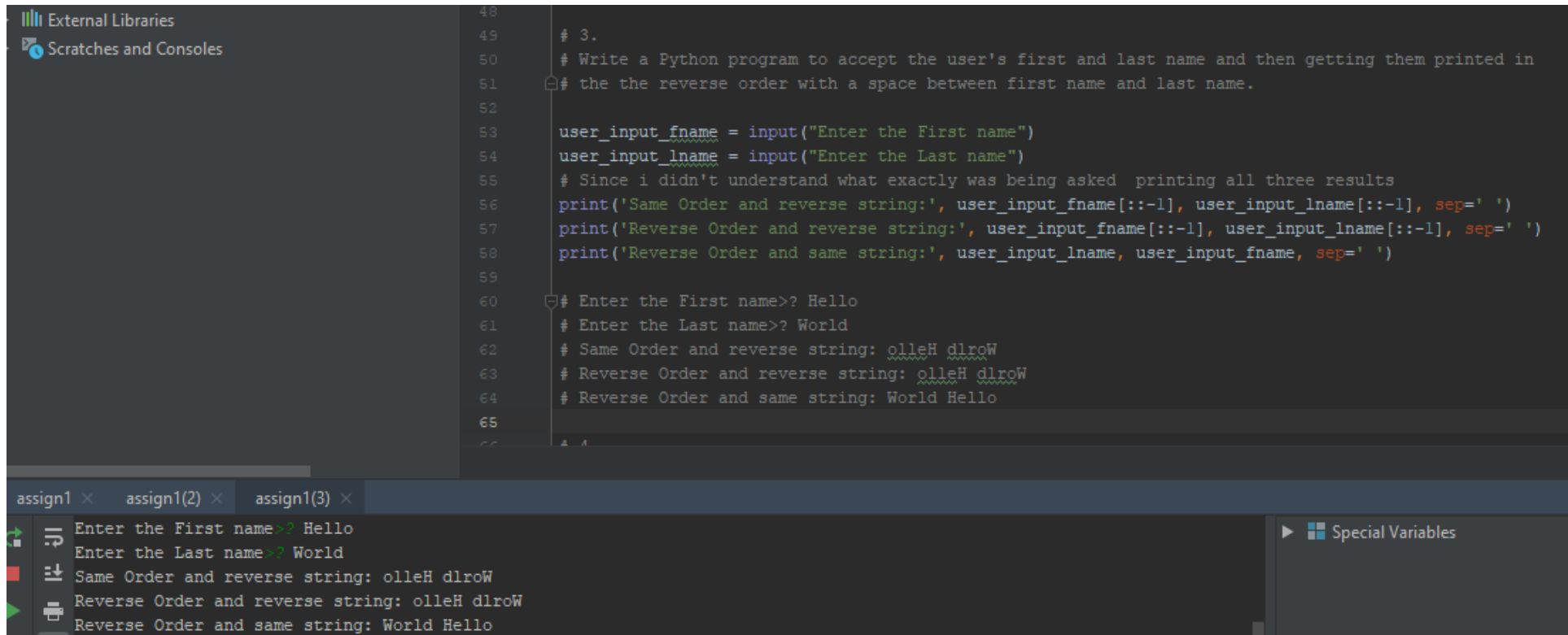
The image shows a PyCharm IDE window with a project named 'ineuron_assign1_hg'. The file 'assign1.py' is open in the editor. The code defines a list 'task1_2_list' and iterates through numbers from 2000 to 3200. It checks if a number is divisible by 7 but not a multiple of 5, and appends it to the list. The output is printed as a single line. The PyDev console shows the execution of the script, displaying the list of numbers that meet the criteria.

```
1 # Author: Harish
2 # Purpose: To finish the first assignment given by Ineuron on 1 May 2020
3
4 # 2.
5 # Write a program which will find all such numbers which are divisible by 7 but are not a multiple
6 # of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a
7 # comma-separated sequence on a single line.
8
9 task1_2_list = []
10 # since we want from 2000 to 3200
11 for i in range(2000, 3201, 1):
12     if i % 5 != 0 and i % 7 == 0:
13         task1_2_list.append(i)
14 else:
15     print("All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7"
16         " but not a multiple of 5 are:\n",
17         task1_2_list)
18
19 # PyDev console: starting.
20 # Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
21 # runfile('...', '/PycharmProjects/ineuron_assign1_hg/assign1.py',
22 # wdir='.../PycharmProjects/ineuron_assign1_hg')
23 # All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7 but not a multiple of 5 are
24 # [2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114, 2121, 2128, 2142, 2149, 2156,
25 # 2163, 2177, 2184, 2191, 2198, 2212, 2219, 2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324,
26 # 2331, 2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436, 2443, 2457, 2464, 2471, 2478, 2492,
27 # 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562, 2569, 2576, 2583, 2597, 2604, 2611, 2618, 2632, 2639, 2646, 2653,
28 # 2667, 2674, 2681, 2688, 2702, 2709, 2716, 2723, 2737, 2744, 2751, 2758, 2772, 2779, 2786, 2793, 2807, 2814, 2821,
29 # 2828, 2842, 2849, 2856, 2863, 2877, 2884, 2891, 2898, 2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 2989,
30 # 2996, 3003, 3017, 3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094, 3101, 3108, 3122, 3129, 3136, 3143, 3157,
31 # 3164, 3171, 3178, 3192, 3199]
```

Approach 2: Using Join method and displaying output as just a string with commas

```
assign1.py x
1 # Author: Harish
2 # Purpose: To finish the first assignment given by Ineuron on 1 May 2020
3
4 # 2.
5 # Write a program which will find all such numbers which are divisible by 7 but are not a multiple
6 # of 5, between 2000 and 3200 (both included). The numbers obtained should be printed in a
7 # comma-separated sequence on a single line.
8
9 task1_2_list = []
10 # since we want from 2000 to 3200
11 for i in range(2000, 3201, 1):
12     if i % 5 != 0 and i % 7 == 0:
13         # converting the integer to string because we want to print the list of numbers without [] and comma separated
14         task1_2_list.append(str(i))
15     else:
16         print("All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7"
17             " but not a multiple of 5 are:\n",
18             ', '.join(task1_2_list)) # used join function with comma separator to print the
19
20 # PyDev console: starting.
21 # Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
22 # runfile('C:/Users/Harish/PycharmProjects/ineuron_assign1_hg/assign1.py',
23 # wdir='C:/Users/Harish/PycharmProjects/ineuron_assign1_hg')
24 # All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7 but not a multiple of 5 are:
25 # 2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114, 2121, 2128, 2142, 2149, 2156,
26 # 2163, 2177, 2184, 2191, 2198, 2212, 2219, 2226, 2233, 2247, 2254, 2261, 2268, 2282, 2289, 2296, 2303, 2317, 2324,
27 # 2331, 2338, 2352, 2359, 2366, 2373, 2387, 2394, 2401, 2408, 2422, 2429, 2436, 2443, 2457, 2464, 2471, 2478, 2492,
28 # 2499, 2506, 2513, 2527, 2534, 2541, 2548, 2562, 2569, 2576, 2583, 2597, 2604, 2611, 2618, 2632, 2639, 2646, 2653,
29 # 2667, 2674, 2681, 2688, 2702, 2709, 2716, 2723, 2737, 2744, 2751, 2758, 2772, 2779, 2786, 2793, 2807, 2814, 2821,
30 # 2828, 2842, 2849, 2856, 2863, 2877, 2884, 2891, 2898, 2912, 2919, 2926, 2933, 2947, 2954, 2961, 2968, 2982, 2989,
31 # 2996, 3003, 3017, 3024, 3031, 3038, 3052, 3059, 3066, 3073, 3087, 3094, 3101, 3108, 3122, 3129, 3136, 3143, 3157,
32 # 3164, 3171, 3178, 3192, 3199
33
```

Task1 :3



The image shows a Python IDE with a dark theme. The top-left sidebar contains 'External Libraries' and 'Scratches and Consoles'. The main editor area displays a Python script with line numbers 48 to 65. The script prompts the user for a first and last name, then prints three different string combinations: the original strings, the strings reversed, and the strings in reverse order. The console at the bottom shows the execution output, including the user's input 'Hello' and 'World', and the resulting printed strings. The tabs at the bottom are labeled 'assign1', 'assign1(2)', and 'assign1(3)'. A 'Special Variables' panel is visible on the right side of the console.

```
48
49 # 3.
50 # Write a Python program to accept the user's first and last name and then getting them printed in
51 # the the reverse order with a space between first name and last name.
52
53 user_input_fname = input("Enter the First name")
54 user_input_lname = input("Enter the Last name")
55 # Since i didn't understand what exactly was being asked printing all three results
56 print('Same Order and reverse string:', user_input_fname[::-1], user_input_lname[::-1], sep=' ')
57 print('Reverse Order and reverse string:', user_input_fname[::-1], user_input_lname[::-1], sep=' ')
58 print('Reverse Order and same string:', user_input_lname, user_input_fname, sep=' ')
59
60 # Enter the First name?? Hello
61 # Enter the Last name?? World
62 # Same Order and reverse string: olleH dlroW
63 # Reverse Order and reverse string: olleH dlroW
64 # Reverse Order and same string: World Hello
65
```

assign1 × assign1(2) × assign1(3) ×

Enter the First name?? Hello
Enter the Last name?? World
Same Order and reverse string: olleH dlroW
Reverse Order and reverse string: olleH dlroW
Reverse Order and same string: World Hello

► Special Variables

Task1:4

The image shows a PyCharm IDE with a Python script named `assign1.py`. The script calculates the volume of a sphere based on user input. The code is as follows:

```
48 # Formula: V=4/3 * π * r³
49 import math
50
51 try:
52     user_input_dia = int(input("Enter the Diameter of the sphere in cm: "))
53 except ValueError:
54     user_input_dia = 12
55 vol = (4/3) * math.pi * pow(user_input_dia / 2, 3)
56
57 print("Volume of a sphere with diameter ", user_input_dia, " cm is : ", round(vol, 3), sep='')
58
59 # The user didn't enter any value and hence it took the default value of 12
60 # Enter the Diameter of the sphere in cm: >?
61 # Volume of a sphere with diameter 12 cm is : 904.779
```

The console output shows the program execution:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
>>> runfile('C:/Users/oyeBuBBly.000/PycharmProjects/ineuron_assign1_hg/assign1.py', wdir='C:/Users/oyeBuBBly.000/PycharmProjects/ineuron_assign1_hg')
All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7 but not a multiple of 5 are:
2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114, 2121, 2128, 2142, 2149, 2156, 2163,
Enter the First name >? Hello
Enter the Last name >? World
olleH dlroW
Enter the Diameter of the sphere in cm: >?
Volume of a sphere with diameter 12 cm is : 904.779
```

The right sidebar shows the Special Variables section with the following variables:

- `i = (int) 3200`
- `task1_2_list = (list) <class 'list'>: ['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022', '2023', '2024', '2025', '2026', '2027', '2028', '2029', '2030', '2031', '2032', '2033', '2034', '2035', '2036', '2037', '2038', '2039', '2040', '2041', '2042', '2043', '2044', '2045', '2046', '2047', '2048', '2049', '2050', '2051', '2052', '2053', '2054', '2055', '2056', '2057', '2058', '2059', '2060', '2061', '2062', '2063', '2064', '2065', '2066', '2067', '2068', '2069', '2070', '2071', '2072', '2073', '2074', '2075', '2076', '2077', '2078', '2079', '2080', '2081', '2082', '2083', '2084', '2085', '2086', '2087', '2088', '2089', '2090', '2091', '2092', '2093', '2094', '2095', '2096', '2097', '2098', '2099', '2100', '2101', '2102', '2103', '2104', '2105', '2106', '2107', '2108', '2109', '2110', '2111', '2112', '2113', '2114', '2115', '2116', '2117', '2118', '2119', '2120', '2121', '2122', '2123', '2124', '2125', '2126', '2127', '2128', '2129', '2130', '2131', '2132', '2133', '2134', '2135', '2136', '2137', '2138', '2139', '2140', '2141', '2142', '2143', '2144', '2145', '2146', '2147', '2148', '2149', '2150', '2151', '2152', '2153', '2154', '2155', '2156', '2157', '2158', '2159', '2160', '2161', '2162', '2163', '2164', '2165', '2166', '2167', '2168', '2169', '2170', '2171', '2172', '2173', '2174', '2175', '2176', '2177', '2178', '2179', '2180', '2181', '2182', '2183', '2184', '2185', '2186', '2187', '2188', '2189', '2190', '2191', '2192', '2193', '2194', '2195', '2196', '2197', '2198', '2199', '2200', '2201', '2202', '2203', '2204', '2205', '2206', '2207', '2208', '2209', '2210', '2211', '2212', '2213', '2214', '2215', '2216', '2217', '2218', '2219', '2220', '2221', '2222', '2223', '2224', '2225', '2226', '2227', '2228', '2229', '2230', '2231', '2232', '2233', '2234', '2235', '2236', '2237', '2238', '2239', '2240', '2241', '2242', '2243', '2244', '2245', '2246', '2247', '2248', '2249', '2250', '2251', '2252', '2253', '2254', '2255', '2256', '2257', '2258', '2259', '2260', '2261', '2262', '2263', '2264', '2265', '2266', '2267', '2268', '2269', '2270', '2271', '2272', '2273', '2274', '2275', '2276', '2277', '2278', '2279', '2280', '2281', '2282', '2283', '2284', '2285', '2286', '2287', '2288', '2289', '2290', '2291', '2292', '2293', '2294', '2295', '2296', '2297', '2298', '2299', '2300', '2301', '2302', '2303', '2304', '2305', '2306', '2307', '2308', '2309', '2310', '2311', '2312', '2313', '2314', '2315', '2316', '2317', '2318', '2319', '2320', '2321', '2322', '2323', '2324', '2325', '2326', '2327', '2328', '2329', '2330', '2331', '2332', '2333', '2334', '2335', '2336', '2337', '2338', '2339', '2340', '2341', '2342', '2343', '2344', '2345', '2346', '2347', '2348', '2349', '2350', '2351', '2352', '2353', '2354', '2355', '2356', '2357', '2358', '2359', '2360', '2361', '2362', '2363', '2364', '2365', '2366', '2367', '2368', '2369', '2370', '2371', '2372', '2373', '2374', '2375', '2376', '2377', '2378', '2379', '2380', '2381', '2382', '2383', '2384', '2385', '2386', '2387', '2388', '2389', '2390', '2391', '2392', '2393', '2394', '2395', '2396', '2397', '2398', '2399', '2400', '2401', '2402', '2403', '2404', '2405', '2406', '2407', '2408', '2409', '2410', '2411', '2412', '2413', '2414', '2415', '2416', '2417', '2418', '2419', '2420', '2421', '2422', '2423', '2424', '2425', '2426', '2427', '2428', '2429', '2430', '2431', '2432', '2433', '2434', '2435', '2436', '2437', '2438', '2439', '2440', '2441', '2442', '2443', '2444', '2445', '2446', '2447', '2448', '2449', '2450', '2451', '2452', '2453', '2454', '2455', '2456', '2457', '2458', '2459', '2460', '2461', '2462', '2463', '2464', '2465', '2466', '2467', '2468', '2469', '2470', '2471', '2472', '2473', '2474', '2475', '2476', '2477', '2478', '2479', '2480', '2481', '2482', '2483', '2484', '2485', '2486', '2487', '2488', '2489', '2490', '2491', '2492', '2493', '2494', '2495', '2496', '2497', '2498', '2499', '2500', '2501', '2502', '2503', '2504', '2505', '2506', '2507', '2508', '2509', '2510', '2511', '2512', '2513', '2514', '2515', '2516', '2517', '2518', '2519', '2520', '2521', '2522', '2523', '2524', '2525', '2526', '2527', '2528', '2529', '2530', '2531', '2532', '2533', '2534', '2535', '2536', '2537', '2538', '2539', '2540', '2541', '2542', '2543', '2544', '2545', '2546', '2547', '2548', '2549', '2550', '2551', '2552', '2553', '2554', '2555', '2556', '2557', '2558', '2559', '2560', '2561', '2562', '2563', '2564', '2565', '2566', '2567', '2568', '2569', '2570', '2571', '2572', '2573', '2574', '2575', '2576', '2577', '2578', '2579', '2580', '2581', '2582', '2583', '2584', '2585', '2586', '2587', '2588', '2589', '2590', '2591', '2592', '2593', '2594', '2595', '2596', '2597', '2598', '2599', '2600', '2601', '2602', '2603', '2604', '2605', '2606', '2607', '2608', '2609', '2610', '2611', '2612', '2613', '2614', '2615', '2616', '2617', '2618', '2619', '2620', '2621', '2622', '2623', '2624', '2625', '2626', '2627', '2628', '2629', '2630', '2631', '2632', '2633', '2634', '2635', '2636', '2637', '2638', '2639', '2640', '2641', '2642', '2643', '2644', '2645', '2646', '2647', '2648', '2649', '2650', '2651', '2652', '2653', '2654', '2655', '2656', '2657', '2658', '2659', '2660', '2661', '2662', '2663', '2664', '2665', '2666', '2667', '2668', '2669', '2670', '2671', '2672', '2673', '2674', '2675', '2676', '2677', '2678', '2679', '2680', '2681', '2682', '2683', '2684', '2685', '2686', '2687', '2688', '2689', '2690', '2691', '2692', '2693', '2694', '2695', '2696', '2697', '2698', '2699', '2700', '2701', '2702', '2703', '2704', '2705', '2706', '2707', '2708', '2709', '2710', '2711', '2712', '2713', '2714', '2715', '2716', '2717', '2718', '2719', '2720', '2721', '2722', '2723', '2724', '2725', '2726', '2727', '2728', '2729', '2730', '2731', '2732', '2733', '2734', '2735', '2736', '2737', '2738', '2739', '2740', '2741', '2742', '2743', '2744', '2745', '2746', '2747', '2748', '2749', '2750', '2751', '2752', '2753', '2754', '2755', '2756', '2757', '2758', '2759', '2760', '2761', '2762', '2763', '2764', '2765', '2766', '2767', '2768', '2769', '2770', '2771', '2772', '2773', '2774', '2775', '2776', '2777', '2778', '2779', '2780', '2781', '2782', '2783', '2784', '2785', '2786', '2787', '2788', '2789', '2790', '2791', '2792', '2793', '2794', '2795', '2796', '2797', '2798', '2799', '2800', '2801', '2802', '2803', '2804', '2805', '2806', '2807', '2808', '2809', '2810', '2811', '2812', '2813', '2814', '2815', '2816', '2817', '2818', '2819', '2820', '2821', '2822', '2823', '2824', '2825', '2826', '2827', '2828', '2829', '2830', '2831', '2832', '2833', '2834', '2835', '2836', '2837', '2838', '2839', '2840', '2841', '2842', '2843', '2844', '2845', '2846', '2847', '2848', '2849', '2850', '2851', '2852', '2853', '2854', '2855', '2856', '2857', '2858', '2859', '2860', '2861', '2862', '2863', '2864', '2865', '2866', '2867', '2868', '2869', '2870', '2871', '2872', '2873', '2874', '2875', '2876', '2877', '2878', '2879', '2880', '2881', '2882', '2883', '2884', '2885', '2886', '2887', '2888', '2889', '2890', '2891', '2892', '2893', '2894', '2895', '2896', '2897', '2898', '2899', '2900', '2901', '2902', '2903', '2904', '2905', '2906', '2907', '2908', '2909', '2910', '2911', '2912', '2913', '2914', '2915', '2916', '2917', '2918', '2919', '2920', '2921', '2922', '2923', '2924', '2925', '2926', '2927', '2928', '2929', '2930', '2931', '2932', '2933', '2934', '2935', '2936', '2937', '2938', '2939', '2940', '2941', '2942', '2943', '2944', '2945', '2946', '2947', '2948', '2949', '2950', '2951', '2952', '2953', '2954', '2955', '2956', '2957', '2958', '2959', '2960', '2961', '2962', '2963', '2964', '2965', '2966', '2967', '2968', '2969', '2970', '2971', '2972', '2973', '2974', '2975', '2976', '2977', '2978', '2979', '2980', '2981', '2982', '2983', '2984', '2985', '2986', '2987', '2988', '2989', '2990', '2991', '2992', '2993', '2994', '2995', '2996', '2997', '2998', '2999', '3000', '3001', '3002', '3003', '3004', '3005', '3006', '3007', '3008', '3009', '3010', '3011', '3012', '3013', '3014', '3015', '3016', '3017', '3018', '3019', '3020', '3021', '3022', '3023', '3024', '3025', '3026', '3027', '3028', '3029', '3030', '3031', '3032', '3033', '3034', '3035', '3036', '3037', '3038', '3039', '3040', '3041', '3042', '3043', '3044', '3045', '3046', '3047', '3048', '3049', '3050', '3051', '3052', '3053', '3054', '3055', '3056', '3057', '3058', '3059', '3060', '3061', '3062', '3063', '3064', '3065', '3066', '3067', '3068', '3069', '3070', '3071', '3072', '3073', '3074', '3075', '3076', '3077', '3078', '3079', '3080', '3081', '3082', '3083', '3084', '3085', '3086', '3087', '3088', '3089', '3090', '3091', '3092', '3093', '3094', '3095', '3096', '3097', '3098', '3099', '3100', '3101', '3102', '3103', '3104', '3105', '3106', '3107', '3108', '3109', '3110', '3111', '3112', '3113', '3114', '3115', '3116', '3117', '3118', '3119', '3120', '3121', '3122', '3123', '3124', '3125', '3126', '3127', '3128', '3129', '3130', '3131', '3132', '3133', '3134', '3135', '3136', '3137', '3138', '3139', '3140', '3141', '3142', '3143', '3144', '3145', '3146', '3147', '3148', '3149', '3150', '3151', '3152', '3153', '3154', '3155', '3156', '3157', '3158', '3159', '3160', '3161', '3162', '3163', '3164', '3165', '3166', '3167', '3168', '3169', '3170', '3171', '3172', '3173', '3174', '3175', '3176', '3177', '3178', '3179', '3180', '3181', '3182', '3183', '3184', '3185', '3186', '3187', '3188', '3189', '3190', '3191', '3192', '3193', '3194', '3195', '3196', '3197', '3198', '3199', '3200']`
- `user_input_dia = (int) 12`
- `user_input_fname = (str) 'Hello'`
- `user_input_lname = (str) 'World'`
- `vol = (float) 904.7786842338603`

Task2:1

The image shows a Python IDE with a script editor and a console window. The script editor contains the following code:

```
64 # 1.  
65 # Write a program which accepts a sequence of comma-separated numbers from console and  
66 # generate a list  
67  
68 inp_seqn = input("enter a seq of numbers sep by a comma")  
69 print(inp_seqn)  
70 str_list = list(inp_seqn.split(','))  
71 int_list = [int(i) for i in str_list]  
72 print(int_list)  
73  
74 # enter a seq of numbers sep by a comma?? 3,4,5,9,6,8  
75 # 3,4,5,9,6,8  
76 # [3, 4, 5, 9, 6, 8]
```

The console window, titled "assign1", shows the following output:

```
All values from 2000 to 3200 (both included) are traversed and the numbers divisible by 7 but not a multiple of 5 are:  
2002, 2009, 2016, 2023, 2037, 2044, 2051, 2058, 2072, 2079, 2086, 2093, 2107, 2114, 2121, 2128, 2142, 2149, 2156, 2163,  
Enter the First name? Hello  
Enter the Last name? World  
olleH dlroW  
Enter the Diameter of the spehere in cm: >?  
Volume of a spehere with diameter 12 cm is : 904.779  
enter a seq of numbers sep by a comma? 3,4,5,9,6,8  
3,4,5,9,6,8  
[3, 4, 5, 9, 6, 8]  
  
>>>
```

On the right side of the console window, there is a variable explorer showing the following variables:

- Special Variab
- 01 i = {int} 3200
- 01 inp_list = {list}
- 01 inp_seqn = {s
- 01 int_list = {list}
- 01 task1_2_list =
- 01 user_input_di
- 01 user_input_fr
- 01 user_input_In
- 01 vol = {float} 9

Task2:2

```
90
91     asterix_list = ["*"]
92     count = 2
93     for j in range(10):
94         for i in asterix_list:
95             if j <= 5:
96                 print(j*i)
97             else:
98                 print((j-count)*i)
99                 count += 2
100
```

assign1 x

*
**

**
*

Task2:3 ; Used two test cases words were TeSt and AcadGild

```
100
101 # 3.
102 # Write a Python program to reverse a word after accepting the input from the user.
103
104 user_word = input("Enter a word")
105 print(user_word[::-1])
106
107 # Enter a word?? TeSt
108 # tSeT
109
```

```
110 # Enter a word>? AcadGild
111 # dliGdacA
```

assign1 ×

Enter a word>? AcadGild
dliGdacA

Task2:4

```
assign1.py
External Libraries
Scratches and Consoles

113
114 # 4.
115 # Write a Python Program to print the given string in the format specified in the sample output.
116 # WE, THE PEOPLE OF INDIA, having solemnly resolved to constitute India into a
117 # SOVEREIGN, SOCIALIST, SECULAR, DEMOCRATIC REPUBLIC and to secure to all
118 # its citizens
119
120 # WE, THE PEOPLE OF INDIA,
121 #     having solemnly resolved to constitute India into a SOVEREIGN, !
122 #     SOCIALIST, SECULAR, DEMOCRATIC REPUBLIC
123 #     and to secure to all its citizens
124 print("WE, THE PEOPLE OF INDIA,\n\thaving solemnly resolved to constitute India into a SOVEREIGN, ! "
125       "\n\t\tSOCIALIST, SECULAR, DEMOCRATIC REPUBLIC \n\t\t and to secure to all its citizens")
126
```

assign1 × assign1(2) ×

```
****
***
**
*
Enter a word>? TeSt
tSeT
WE, THE PEOPLE OF INDIA,
    having solemnly resolved to constitute India into a SOVEREIGN, !
    SOCIALIST, SECULAR, DEMOCRATIC REPUBLIC
    and to secure to all its citizens
```

Special Variables

- asterix_list = {list} ['**']
- count = {int} 10
- i = {str} '**'
- inp_seqn = {str} '3'
- int_list = {list} [3]
- j = {int} 9
- str_list = {list} ['3']
- task1_2_list = {list} <class 'list':
- user_input_dia = {int} 12