In [1]:

```python
import numpy
import pandas
import statsmodels.api as sm
def simple_heuristic(file_path):
    '''
    In this exercise, we will perform some rudimentary practices similar to those of
    an actual data scientist.

    Part of a data scientist's job is to use her or his intuition and insight to
    write algorithms and heuristics. A data scientist also creates mathematical models
    to make predictions based on some attributes from the data that they are examining.

    We would like for you to take your knowledge and intuition about the Titanic
    and its passengers' attributes to predict whether or not the passengers survived
    or perished. You can read more about the Titanic and specifics about this dataset at:
    http://en.wikipedia.org/wiki/RMS_Titanic
    http://www.kaggle.com/c/titanic-gettingStarted

    In this exercise and the following ones, you are given a list of Titantic passengers
    and their associated information. More information about the data can be seen at the
    link below:
    http://www.kaggle.com/c/titanic-gettingStarted/data.

    For this exercise, you need to write a simple heuristic that will use
    the passengers' gender to predict if that person survived the Titanic disaster.

    You prediction should be 78% accurate or higher.

    Here's a simple heuristic to start off:
       1) If the passenger is female, your heuristic should assume that the
       passenger survived.
       2) If the passenger is male, you heuristic should
       assume that the passenger did not survive.

    You can access the gender of a passenger via passenger['Sex'].
    If the passenger is male, passenger['Sex'] will return a string "male".
    If the passenger is female, passenger['Sex'] will return a string "female".

    Write your prediction back into the "predictions" dictionary. The
    key of the dictionary should be the passenger's id (which can be accessed
    via passenger["PassengerId"]) and the associated value should be 1 if the
    passenger survied or 0 otherwise.

    For example, if a passenger is predicted to have survived:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 1

    And if a passenger is predicted to have perished in the disaster:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 0

    You can also look at the Titantic data that you will be working with
    at the link below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic_data.csv
    '''

    predictions = {}
    df = pandas.read_excel(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

        # Your code here:
        # If the passenger is a female, then the passenger survived otherwise not servived.
        if passenger['Sex'] == 'female':

            predictions[passenger_id] = 1
        elif(passenger['Sex']=='male'):
            predictions[passenger_id] = 0

    return predictions
if __name__ == "__main__":
    file_path=r'Survivor Data Set.xlsx'
    output=simple_heuristic(file_path)
    print(output)
```

```
{1: 0, 2: 1, 3: 1, 4: 1, 5: 0, 6: 0, 7: 0, 8: 0, 9: 1, 10: 1, 11: 1, 12: 1, 13: 0, 14: 0, 15: 1, 16: 1, 17: 0, 18:
0, 19: 1, 20: 1, 21: 0, 22: 0, 23: 1, 24: 0, 25: 1, 26: 1, 27: 0, 28: 0, 29: 1, 30: 0, 31: 0, 32: 1, 33: 1, 34: 0,
35: 0, 36: 0, 37: 0, 38: 0, 39: 1, 40: 1, 41: 1, 42: 1, 43: 0, 44: 1, 45: 1, 46: 0, 47: 0, 48: 1, 49: 0, 50: 1, 51:
0, 52: 0, 53: 1, 54: 1, 55: 0, 56: 0, 57: 1, 58: 0, 59: 1, 60: 0, 61: 0, 62: 1, 63: 0, 64: 0, 65: 0, 66: 0, 67: 1,
68: 0, 69: 1, 70: 0, 71: 0, 72: 1, 73: 0, 74: 0, 75: 0, 76: 0, 77: 0, 78: 0, 79: 0, 80: 1, 81: 0, 82: 0, 83: 1, 84:
0, 85: 1, 86: 1, 87: 0, 88: 0, 89: 1, 90: 0, 91: 0, 92: 0, 93: 0, 94: 0, 95: 0, 96: 0, 97: 0, 98: 0, 99: 1, 100: 0,
101: 1, 102: 0, 103: 0, 104: 0, 105: 0, 106: 0, 107: 1, 108: 0, 109: 0, 110: 1, 111: 0, 112: 1, 113: 0, 114: 1, 11
5: 1, 116: 0, 117: 0, 118: 0, 119: 0, 120: 1, 121: 0, 122: 0, 123: 0, 124: 1, 125: 0, 126: 0, 127: 0, 128: 0, 129:
1, 130: 0, 131: 0, 132: 0, 133: 1, 134: 1, 135: 0, 136: 0, 137: 1, 138: 0, 139: 0, 140: 0, 141: 1, 142: 1, 143: 1,
144: 0, 145: 0, 146: 0, 147: 0, 148: 1, 149: 0, 150: 0, 151: 0, 152: 1, 153: 0, 154: 0, 155: 0, 156: 0, 157: 1, 15
8: 0, 159: 0, 160: 0, 161: 0, 162: 1, 163: 0, 164: 0, 165: 0, 166: 0, 167: 1, 168: 1, 169: 0, 170: 0, 171: 0, 172:
0, 173: 1, 174: 0, 175: 0, 176: 0, 177: 0, 178: 1, 179: 0, 180: 0, 181: 1, 182: 0, 183: 0, 184: 0, 185: 1, 186: 0,
187: 1, 188: 0, 189: 0, 190: 0, 191: 1, 192: 0, 193: 1, 194: 0, 195: 1, 196: 1, 197: 0, 198: 0, 199: 1, 200: 1, 20
1: 0, 202: 0, 203: 0, 204: 0, 205: 0, 206: 1, 207: 0, 208: 0, 209: 1, 210: 0, 211: 0, 212: 1, 213: 0, 214: 0, 215:
0, 216: 1, 217: 1, 218: 0, 219: 1, 220: 0, 221: 0, 222: 0, 223: 0, 224: 0, 225: 0, 226: 0, 227: 0, 228: 0, 229: 0,
230: 1, 231: 1, 232: 0, 233: 0, 234: 1, 235: 0, 236: 1, 237: 0, 238: 1, 239: 0, 240: 0, 241: 1, 242: 1, 243: 0, 24
4: 0, 245: 0, 246: 0, 247: 1, 248: 1, 249: 0, 250: 0, 251: 0, 252: 1, 253: 0, 254: 0, 255: 1, 256: 1, 257: 1, 258:
1, 259: 1, 260: 1, 261: 0, 262: 0, 263: 0, 264: 0, 265: 1, 266: 0, 267: 0, 268: 0, 269: 1, 270: 1, 271: 0, 272: 0,
273: 1, 274: 0, 275: 1, 276: 1, 277: 1, 278: 0, 279: 0, 280: 1, 281: 0, 282: 0, 283: 0, 284: 0, 285: 0, 286: 0, 28
7: 0, 288: 0, 289: 0, 290: 1, 291: 1, 292: 1, 293: 0, 294: 1, 295: 0, 296: 0, 297: 0, 298: 1, 299: 0, 300: 1, 301:
1, 302: 0, 303: 0, 304: 1, 305: 0, 306: 0, 307: 1, 308: 1, 309: 0, 310: 1, 311: 1, 312: 1, 313: 1, 314: 0, 315: 0,
316: 1, 317: 1, 318: 0, 319: 1, 320: 1, 321: 0, 322: 0, 323: 1, 324: 1, 325: 0, 326: 1, 327: 0, 328: 1, 329: 1, 33
0: 1, 331: 1, 332: 0, 333: 0, 334: 0, 335: 1, 336: 0, 337: 0, 338: 1, 339: 0, 340: 0, 341: 0, 342: 1, 343: 0, 344:
0, 345: 0, 346: 1, 347: 1, 348: 1, 349: 0, 350: 0, 351: 0, 352: 0, 353: 0, 354: 0, 355: 0, 356: 0, 357: 1, 358: 1,
359: 1, 360: 1, 361: 0, 362: 0, 363: 1, 364: 0, 365: 0, 366: 0, 367: 1, 368: 1, 369: 1, 370: 1, 371: 0, 372: 0, 37
3: 0, 374: 0, 375: 1, 376: 1, 377: 1, 378: 0, 379: 0, 380: 0, 381: 1, 382: 1, 383: 0, 384: 1, 385: 0, 386: 0, 387:
0, 388: 1, 389: 0, 390: 1, 391: 0, 392: 0, 393: 0, 394: 1, 395: 1, 396: 0, 397: 1, 398: 0, 399: 0, 400: 1, 401: 0,
402: 0, 403: 1, 404: 0, 405: 1, 406: 0, 407: 0, 408: 0, 409: 0, 410: 1, 411: 0, 412: 0, 413: 1, 414: 0, 415: 0, 41
6: 1, 417: 1, 418: 1, 419: 0, 420: 1, 421: 0, 422: 0, 423: 0, 424: 1, 425: 0, 426: 0, 427: 1, 428: 1, 429: 0, 430:
0, 431: 0, 432: 1, 433: 1, 434: 0, 435: 0, 436: 1, 437: 1, 438: 1, 439: 0, 440: 0, 441: 1, 442: 0, 443: 0, 444: 1,
445: 0, 446: 0, 447: 1, 448: 0, 449: 1, 450: 0, 451: 0, 452: 0, 453: 0, 454: 0, 455: 0, 456: 0, 457: 0, 458: 1, 45
9: 1, 460: 0, 461: 0, 462: 0, 463: 0, 464: 0, 465: 0, 466: 0, 467: 0, 468: 0, 469: 0, 470: 1, 471: 0, 472: 0, 473:
1, 474: 1, 475: 1, 476: 0, 477: 0, 478: 0, 479: 0, 480: 1, 481: 0, 482: 0, 483: 0, 484: 1, 485: 0, 486: 1, 487: 1,
488: 0, 489: 0, 490: 0, 491: 0, 492: 0, 493: 0, 494: 0, 495: 0, 496: 0, 497: 1, 498: 0, 499: 1, 500: 0, 501: 0, 50
2: 1, 503: 1, 504: 1, 505: 1, 506: 0, 507: 1, 508: 0, 509: 0, 510: 0, 511: 0, 512: 0, 513: 0, 514: 1, 515: 0, 516:
0, 517: 1, 518: 0, 519: 1, 520: 0, 521: 1, 522: 0, 523: 0, 524: 1, 525: 0, 526: 0, 527: 1, 528: 0, 529: 0, 530: 0,
531: 1, 532: 0, 533: 0, 534: 1, 535: 1, 536: 1, 537: 0, 538: 1, 539: 0, 540: 1, 541: 1, 542: 1, 543: 1, 544: 0, 54
5: 0, 546: 0, 547: 1, 548: 0, 549: 0, 550: 0, 551: 0, 552: 0, 553: 0, 554: 0, 555: 1, 556: 0, 557: 1, 558: 0, 559:
1, 560: 1, 561: 0, 562: 0, 563: 0, 564: 0, 565: 1, 566: 0, 567: 0, 568: 1, 569: 0, 570: 0, 571: 0, 572: 1, 573: 0,
574: 1, 575: 0, 576: 0, 577: 1, 578: 1, 579: 1, 580: 0, 581: 1, 582: 1, 583: 0, 584: 0, 585: 0, 586: 1, 587: 0, 58
8: 0, 589: 0, 590: 0, 591: 0, 592: 1, 593: 0, 594: 1, 595: 0, 596: 0, 597: 1, 598: 0, 599: 0, 600: 0, 601: 1, 602:
0, 603: 0, 604: 0, 605: 0, 606: 0, 607: 0, 608: 0, 609: 1, 610: 1, 611: 1, 612: 0, 613: 1, 614: 0, 615: 0, 616: 1,
617: 0, 618: 1, 619: 1, 620: 0, 621: 0, 622: 0, 623: 0, 624: 0, 625: 0, 626: 0, 627: 0, 628: 1, 629: 0, 630: 0, 63
1: 0, 632: 0, 633: 0, 634: 0, 635: 1, 636: 1, 637: 0, 638: 0, 639: 1, 640: 0, 641: 0, 642: 1, 643: 1, 644: 0, 645:
1, 646: 0, 647: 0, 648: 0, 649: 0, 650: 1, 651: 0, 652: 1, 653: 0, 654: 1, 655: 1, 656: 0, 657: 0, 658: 1, 659: 0,
660: 0, 661: 0, 662: 0, 663: 0, 664: 0, 665: 0, 666: 0, 667: 0, 668: 0, 669: 0, 670: 1, 671: 1, 672: 0, 673: 0, 67
4: 0, 675: 0, 676: 0, 677: 0, 678: 1, 679: 1, 680: 0, 681: 1, 682: 0, 683: 0, 684: 0, 685: 0, 686: 0, 687: 0, 688:
0, 689: 0, 690: 1, 691: 0, 692: 1, 693: 0, 694: 0, 695: 0, 696: 0, 697: 0, 698: 1, 699: 0, 700: 0, 701: 1, 702: 0,
703: 1, 704: 0, 705: 0, 706: 0, 707: 1, 708: 0, 709: 1, 710: 0, 711: 1, 712: 0, 713: 0, 714: 0, 715: 0, 716: 0, 71
7: 1, 718: 1, 719: 0, 720: 0, 721: 1, 722: 0, 723: 0, 724: 0, 725: 0, 726: 0, 727: 1, 728: 1, 729: 0, 730: 1, 731:
1, 732: 0, 733: 0, 734: 0, 735: 0, 736: 0, 737: 1, 738: 0, 739: 0, 740: 0, 741: 0, 742: 0, 743: 1, 744: 0, 745: 0,
746: 0, 747: 0, 748: 1, 749: 0, 750: 0, 751: 1, 752: 0, 753: 0, 754: 0, 755: 1, 756: 0, 757: 0, 758: 0, 759: 0, 76
0: 1, 761: 0, 762: 0, 763: 0, 764: 1, 765: 0, 766: 1, 767: 0, 768: 1, 769: 0, 770: 0, 771: 0, 772: 0, 773: 1, 774:
0, 775: 1, 776: 0, 777: 0, 778: 1, 779: 0, 780: 1, 781: 1, 782: 1, 783: 0, 784: 0, 785: 0, 786: 0, 787: 1, 788: 0,
789: 0, 790: 0, 791: 0, 792: 0, 793: 1, 794: 0, 795: 0, 796: 0, 797: 1, 798: 1, 799: 0, 800: 1, 801: 0, 802: 1, 80
3: 0, 804: 0, 805: 0, 806: 0, 807: 0, 808: 1, 809: 0, 810: 1, 811: 0, 812: 0, 813: 0, 814: 1, 815: 0, 816: 0, 817:
1, 818: 0, 819: 0, 820: 0, 821: 1, 822: 0, 823: 0, 824: 1, 825: 0, 826: 0, 827: 0, 828: 0, 829: 0, 830: 1, 831: 1,
832: 0, 833: 0, 834: 0, 835: 0, 836: 1, 837: 0, 838: 0, 839: 0, 840: 0, 841: 0, 842: 0, 843: 1, 844: 0, 845: 0, 84
6: 0, 847: 0, 848: 0, 849: 0, 850: 1, 851: 0, 852: 0, 853: 1, 854: 1, 855: 1, 856: 1, 857: 1, 858: 0, 859: 1, 860:
0, 861: 0, 862: 0, 863: 1, 864: 1, 865: 0, 866: 1, 867: 1, 868: 0, 869: 0, 870: 0, 871: 0, 872: 1, 873: 0, 874: 0,
875: 1, 876: 1, 877: 0, 878: 0, 879: 0, 880: 1, 881: 1, 882: 0, 883: 1, 884: 0, 885: 0, 886: 1, 887: 0, 888: 1, 88
9: 1, 890: 0, 891: 0}
```

In [7]:

```
import numpy
import pandas
import statsmodels.api as sm

def complex_heuristic(file_path):
    '''
    You are given a list of Titantic passengers and their associated
    information. More information about the data can be seen at the link below:
    http://www.kaggle.com/c/titanic-gettingStarted/data

    For this exercise, you need to write a more sophisticated algorithm
    that will use the passengers' gender and their socioeconomical class and age
    to predict if they survived the Titanic disaster.

    You prediction should be 79% accurate or higher.

    Here's the algorithm, predict the passenger survived if:
    1) If the passenger is female or
    2) if his/her socioeconomic status is high AND if the passenger is under 18

    Otherwise, your algorithm should predict that the passenger perished in the disaster.

    Or more specifically in terms of coding:
    female or (high status and under 18)

    You can access the gender of a passenger via passenger['Sex'].
    If the passenger is male, passenger['Sex'] will return a string "male".
    If the passenger is female, passenger['Sex'] will return a string "female".

    You can access the socioeconomic status of a passenger via passenger['Pclass']:
    High socioeconomic status -- passenger['Pclass'] is 1
    Medium socioeconomic status -- passenger['Pclass'] is 2
    Low socioeconomic status -- passenger['Pclass'] is 3

    You can access the age of a passenger via passenger['Age'].

    Write your prediction back into the "predictions" dictionary. The
    key of the dictionary should be the Passenger's id (which can be accessed
    via passenger["PassengerId"]) and the associated value should be 1 if the
    passenger survived or 0 otherwise.

    For example, if a passenger is predicted to have survived:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 1

    And if a passenger is predicted to have perished in the disaster:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 0

    You can also look at the Titantic data that you will be working with
    at the link below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic_data.csv
    '''
    predictions = {}
    df = pandas.read_excel(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

# Here's the algorithm, predict the passenger survived
# 1) If the passenger is female or
# 2) if his/her socioeconomic status is high AND if the passenger is under 18
# Otherwise, your algorithm should predict that the passenger perished in the disaster
        if passenger['Sex'] == 'female' or (passenger['Pclass'] ==1 and passenger['Age'] < 18):
            predictions[passenger_id] = 1
        else:
            predictions[passenger_id] = 0
    return predictions
if __name__ == "__main__":
    file_path=r'Survivor Data Set.xlsx'
    output=complex_heuristic(file_path)
    print(output)
```

{1: 0, 2: 1, 3: 1, 4: 1, 5: 0, 6: 0, 7: 0, 8: 0, 9: 1, 10: 1, 11: 1, 12: 1, 13: 0, 14: 0, 15: 1, 16: 1, 17: 0, 18:
0, 19: 1, 20: 1, 21: 0, 22: 0, 23: 1, 24: 0, 25: 1, 26: 1, 27: 0, 28: 0, 29: 1, 30: 0, 31: 0, 32: 1, 33: 1, 34: 0,
35: 0, 36: 0, 37: 0, 38: 0, 39: 1, 40: 1, 41: 1, 42: 1, 43: 0, 44: 1, 45: 1, 46: 0, 47: 0, 48: 1, 49: 0, 50: 1, 51:
0, 52: 0, 53: 1, 54: 1, 55: 0, 56: 0, 57: 1, 58: 0, 59: 1, 60: 0, 61: 0, 62: 1, 63: 0, 64: 0, 65: 0, 66: 0, 67: 1,
68: 0, 69: 1, 70: 0, 71: 0, 72: 1, 73: 0, 74: 0, 75: 0, 76: 0, 77: 0, 78: 0, 79: 0, 80: 1, 81: 0, 82: 0, 83: 1, 84:
0, 85: 1, 86: 1, 87: 0, 88: 0, 89: 1, 90: 0, 91: 0, 92: 0, 93: 0, 94: 0, 95: 0, 96: 0, 97: 0, 98: 0, 99: 1, 100: 0,
101: 1, 102: 0, 103: 0, 104: 0, 105: 0, 106: 0, 107: 1, 108: 0, 109: 0, 110: 1, 111: 0, 112: 1, 113: 0, 114: 1, 11
5: 1, 116: 0, 117: 0, 118: 0, 119: 0, 120: 1, 121: 0, 122: 0, 123: 0, 124: 1, 125: 0, 126: 0, 127: 0, 128: 0, 129:
1, 130: 0, 131: 0, 132: 0, 133: 1, 134: 1, 135: 0, 136: 0, 137: 1, 138: 0, 139: 0, 140: 0, 141: 1, 142: 1, 143: 1,
144: 0, 145: 0, 146: 0, 147: 0, 148: 1, 149: 0, 150: 0, 151: 0, 152: 1, 153: 0, 154: 0, 155: 0, 156: 0, 157: 1, 15
8: 0, 159: 0, 160: 0, 161: 0, 162: 1, 163: 0, 164: 0, 165: 0, 166: 0, 167: 1, 168: 1, 169: 0, 170: 0, 171: 0, 172:
0, 173: 1, 174: 0, 175: 0, 176: 0, 177: 0, 178: 1, 179: 0, 180: 0, 181: 1, 182: 0, 183: 0, 184: 0, 185: 1, 186: 0,
187: 1, 188: 0, 189: 0, 190: 0, 191: 1, 192: 0, 193: 1, 194: 0, 195: 1, 196: 1, 197: 0, 198: 0, 199: 1, 200: 1, 20
1: 0, 202: 0, 203: 0, 204: 0, 205: 0, 206: 1, 207: 0, 208: 0, 209: 1, 210: 0, 211: 0, 212: 1, 213: 0, 214: 0, 215:
0, 216: 1, 217: 1, 218: 0, 219: 1, 220: 0, 221: 0, 222: 0, 223: 0, 224: 0, 225: 0, 226: 0, 227: 0, 228: 0, 229: 0,
230: 1, 231: 1, 232: 0, 233: 0, 234: 1, 235: 0, 236: 1, 237: 0, 238: 1, 239: 0, 240: 0, 241: 1, 242: 1, 243: 0, 24
4: 0, 245: 0, 246: 0, 247: 1, 248: 1, 249: 0, 250: 0, 251: 0, 252: 1, 253: 0, 254: 0, 255: 1, 256: 1, 257: 1, 258:
1, 259: 1, 260: 1, 261: 0, 262: 0, 263: 0, 264: 0, 265: 1, 266: 0, 267: 0, 268: 0, 269: 1, 270: 1, 271: 0, 272: 0,
273: 1, 274: 0, 275: 1, 276: 1, 277: 1, 278: 0, 279: 0, 280: 1, 281: 0, 282: 0, 283: 0, 284: 0, 285: 0, 286: 0, 28
7: 0, 288: 0, 289: 0, 290: 1, 291: 1, 292: 1, 293: 0, 294: 1, 295: 0, 296: 0, 297: 0, 298: 1, 299: 0, 300: 1, 301:
1, 302: 0, 303: 0, 304: 1, 305: 0, 306: 1, 307: 1, 308: 1, 309: 0, 310: 1, 311: 1, 312: 1, 313: 1, 314: 0, 315: 0,
316: 1, 317: 1, 318: 0, 319: 1, 320: 1, 321: 0, 322: 0, 323: 1, 324: 1, 325: 0, 326: 1, 327: 0, 328: 1, 329: 1, 33
0: 1, 331: 1, 332: 0, 333: 0, 334: 0, 335: 1, 336: 0, 337: 0, 338: 1, 339: 0, 340: 0, 341: 0, 342: 1, 343: 0, 344:
0, 345: 0, 346: 1, 347: 1, 348: 1, 349: 0, 350: 0, 351: 0, 352: 0, 353: 0, 354: 0, 355: 0, 356: 0, 357: 1, 358: 1,
359: 1, 360: 1, 361: 0, 362: 0, 363: 1, 364: 0, 365: 0, 366: 0, 367: 1, 368: 1, 369: 1, 370: 1, 371: 0, 372: 0, 37
3: 0, 374: 0, 375: 1, 376: 1, 377: 1, 378: 0, 379: 0, 380: 0, 381: 1, 382: 1, 383: 0, 384: 1, 385: 0, 386: 0, 387:
0, 388: 1, 389: 0, 390: 1, 391: 0, 392: 0, 393: 0, 394: 1, 395: 1, 396: 0, 397: 1, 398: 0, 399: 0, 400: 1, 401: 0,
402: 0, 403: 1, 404: 0, 405: 1, 406: 0, 407: 0, 408: 0, 409: 0, 410: 1, 411: 0, 412: 0, 413: 1, 414: 0, 415: 0, 41
6: 1, 417: 1, 418: 1, 419: 0, 420: 1, 421: 0, 422: 0, 423: 0, 424: 1, 425: 0, 426: 0, 427: 1, 428: 1, 429: 0, 430:
0, 431: 0, 432: 1, 433: 1, 434: 0, 435: 0, 436: 1, 437: 1, 438: 1, 439: 0, 440: 0, 441: 1, 442: 0, 443: 0, 444: 1,
445: 0, 446: 1, 447: 1, 448: 0, 449: 1, 450: 0, 451: 0, 452: 0, 453: 0, 454: 0, 455: 0, 456: 0, 457: 0, 458: 1, 45
9: 1, 460: 0, 461: 0, 462: 0, 463: 0, 464: 0, 465: 0, 466: 0, 467: 0, 468: 0, 469: 0, 470: 1, 471: 0, 472: 0, 473:
1, 474: 1, 475: 1, 476: 0, 477: 0, 478: 0, 479: 0, 480: 1, 481: 0, 482: 0, 483: 0, 484: 1, 485: 0, 486: 1, 487: 1,
488: 0, 489: 0, 490: 0, 491: 0, 492: 0, 493: 0, 494: 0, 495: 0, 496: 0, 497: 1, 498: 0, 499: 1, 500: 0, 501: 0, 50
2: 1, 503: 1, 504: 1, 505: 1, 506: 0, 507: 1, 508: 0, 509: 0, 510: 0, 511: 0, 512: 0, 513: 0, 514: 1, 515: 0, 516:
0, 517: 1, 518: 0, 519: 1, 520: 0, 521: 1, 522: 0, 523: 0, 524: 1, 525: 0, 526: 0, 527: 1, 528: 0, 529: 0, 530: 0,
531: 1, 532: 0, 533: 0, 534: 1, 535: 1, 536: 1, 537: 0, 538: 1, 539: 0, 540: 1, 541: 1, 542: 1, 543: 1, 544: 0, 54
5: 0, 546: 0, 547: 1, 548: 0, 549: 0, 550: 0, 551: 1, 552: 0, 553: 0, 554: 0, 555: 1, 556: 0, 557: 1, 558: 0, 559:
1, 560: 1, 561: 0, 562: 0, 563: 0, 564: 0, 565: 1, 566: 0, 567: 0, 568: 1, 569: 0, 570: 0, 571: 0, 572: 1, 573: 0,
574: 1, 575: 0, 576: 0, 577: 1, 578: 1, 579: 1, 580: 0, 581: 1, 582: 1, 583: 0, 584: 0, 585: 0, 586: 1, 587: 0, 58
8: 0, 589: 0, 590: 0, 591: 0, 592: 1, 593: 0, 594: 1, 595: 0, 596: 0, 597: 1, 598: 0, 599: 0, 600: 0, 601: 1, 602:
0, 603: 0, 604: 0, 605: 0, 606: 0, 607: 0, 608: 0, 609: 1, 610: 1, 611: 1, 612: 0, 613: 1, 614: 0, 615: 0, 616: 1,
617: 0, 618: 1, 619: 1, 620: 0, 621: 0, 622: 0, 623: 0, 624: 0, 625: 0, 626: 0, 627: 0, 628: 1, 629: 0, 630: 0, 63
1: 0, 632: 0, 633: 0, 634: 0, 635: 1, 636: 1, 637: 0, 638: 0, 639: 1, 640: 0, 641: 0, 642: 1, 643: 1, 644: 0, 645:
1, 646: 0, 647: 0, 648: 0, 649: 0, 650: 1, 651: 0, 652: 1, 653: 0, 654: 1, 655: 1, 656: 0, 657: 0, 658: 1, 659: 0,
660: 0, 661: 0, 662: 0, 663: 0, 664: 0, 665: 0, 666: 0, 667: 0, 668: 0, 669: 0, 670: 1, 671: 1, 672: 0, 673: 0, 67
4: 0, 675: 0, 676: 0, 677: 0, 678: 1, 679: 1, 680: 0, 681: 1, 682: 0, 683: 0, 684: 0, 685: 0, 686: 0, 687: 0, 688:
0, 689: 0, 690: 1, 691: 0, 692: 1, 693: 0, 694: 0, 695: 0, 696: 0, 697: 0, 698: 1, 699: 0, 700: 0, 701: 1, 702: 0,
703: 1, 704: 0, 705: 0, 706: 0, 707: 1, 708: 0, 709: 1, 710: 0, 711: 1, 712: 0, 713: 0, 714: 0, 715: 0, 716: 0, 71
7: 1, 718: 1, 719: 0, 720: 0, 721: 1, 722: 0, 723: 0, 724: 0, 725: 0, 726: 0, 727: 1, 728: 1, 729: 0, 730: 1, 731:
1, 732: 0, 733: 0, 734: 0, 735: 0, 736: 0, 737: 1, 738: 0, 739: 0, 740: 0, 741: 0, 742: 0, 743: 1, 744: 0, 745: 0,
746: 0, 747: 0, 748: 1, 749: 0, 750: 0, 751: 1, 752: 0, 753: 0, 754: 0, 755: 1, 756: 0, 757: 0, 758: 0, 759: 0, 76
0: 1, 761: 0, 762: 0, 763: 0, 764: 1, 765: 0, 766: 1, 767: 0, 768: 1, 769: 0, 770: 0, 771: 0, 772: 0, 773: 1, 774:
0, 775: 1, 776: 0, 777: 0, 778: 1, 779: 0, 780: 1, 781: 1, 782: 1, 783: 0, 784: 0, 785: 0, 786: 0, 787: 1, 788: 0,
789: 0, 790: 0, 791: 0, 792: 0, 793: 1, 794: 0, 795: 0, 796: 0, 797: 1, 798: 1, 799: 0, 800: 1, 801: 0, 802: 1, 80
3: 1, 804: 0, 805: 0, 806: 0, 807: 0, 808: 1, 809: 0, 810: 1, 811: 0, 812: 0, 813: 0, 814: 1, 815: 0, 816: 0, 817:
1, 818: 0, 819: 0, 820: 0, 821: 1, 822: 0, 823: 0, 824: 1, 825: 0, 826: 0, 827: 0, 828: 0, 829: 0, 830: 1, 831: 1,
832: 0, 833: 0, 834: 0, 835: 0, 836: 1, 837: 0, 838: 0, 839: 0, 840: 0, 841: 0, 842: 0, 843: 1, 844: 0, 845: 0, 84
6: 0, 847: 0, 848: 0, 849: 0, 850: 1, 851: 0, 852: 0, 853: 1, 854: 1, 855: 1, 856: 1, 857: 1, 858: 0, 859: 1, 860:
0, 861: 0, 862: 0, 863: 1, 864: 1, 865: 0, 866: 1, 867: 1, 868: 0, 869: 0, 870: 0, 871: 0, 872: 1, 873: 0, 874: 0,
875: 1, 876: 1, 877: 0, 878: 0, 879: 0, 880: 1, 881: 1, 882: 0, 883: 1, 884: 0, 885: 0, 886: 1, 887: 0, 888: 1, 88
9: 1, 890: 0, 891: 0}

In [12]:

```python
import numpy
import pandas
import statsmodels.api as sm

def custom_heuristic(file_path):
    '''
    You are given a list of Titantic passengers and their associated
    information. More information about the data can be seen at the link below:
    http://www.kaggle.com/c/titanic-gettingStarted/data

    For this exercise, you need to write a custom heuristic that will take
    in some combination of the passenger's attributes and predict if the passenger
    survived the Titanic diaster.

    Can your custom heuristic beat 80% accuracy?

    The available attributes are:
    Pclass          Passenger Class
                    (1 = 1st; 2 = 2nd; 3 = 3rd)
    Name            Name
    Sex             Sex
    Age             Age
    SibSp           Number of Siblings/Spouses Aboard
    Parch           Number of Parents/Children Aboard
    Ticket          Ticket Number
    Fare            Passenger Fare
    Cabin           Cabin
    Embarked        Port of Embarkation
                    (C = Cherbourg; Q = Queenstown; S = Southampton)

    SPECIAL NOTES:
    Pclass is a proxy for socioeconomic status (SES)
    1st ~ Upper; 2nd ~ Middle; 3rd ~ Lower

    Age is in years; fractional if age less than one
    If the age is estimated, it is in the form xx.5

    With respect to the family relation variables (i.e. SibSp and Parch)
    some relations were ignored. The following are the definitions used
    for SibSp and Parch.

    Sibling:  brother, sister, stepbrother, or stepsister of passenger aboard Titanic
    Spouse:   husband or wife of passenger aboard Titanic (mistresses and fiancees ignored)
    Parent:   mother or father of passenger aboard Titanic
    Child:    son, daughter, stepson, or stepdaughter of passenger aboard Titanic

    Write your prediction back into the "predictions" dictionary. The
    key of the dictionary should be the passenger's id (which can be accessed
    via passenger["PassengerId"]) and the associating value should be 1 if the
    passenger survvied or 0 otherwise.

    For example, if a passenger is predicted to have survived:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 1

    And if a passenger is predicted to have perished in the disaster:
    passenger_id = passenger['PassengerId']
    predictions[passenger_id] = 0

    You can also look at the Titantic data that you will be working with
    at the link below:
    https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/titanic_data.csv
    '''

    predictions = {}
    df = pandas.read_excel(file_path)
    for passenger_index, passenger in df.iterrows():
        passenger_id = passenger['PassengerId']

        # default assumption: the passenger perished
        predictions[passenger_id] = 0

        # assume that all women and children not in 3rd class survived
        if (passenger['Sex']=='female' or passenger['Age'] < 15) and passenger['Pclass'] != 3:
            predictions[passenger_id] = 1

    return predictions


if __name__ == "__main__":
    file_path=r'Survivor Data Set.xlsx'
    output=custom_heuristic(file_path)
    print(output)
{1: 0, 2: 1, 3: 0, 4: 1, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 1, 11: 0, 12: 1, 13: 0, 14: 0, 15: 0, 16: 1, 17: 0, 18:
0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 0, 29: 0, 30: 0, 31: 0, 32: 1, 33: 0, 34: 0,
35: 0, 36: 0, 37: 0, 38: 0, 39: 0, 40: 0, 41: 0, 42: 1, 43: 0, 44: 1, 45: 0, 46: 0, 47: 0, 48: 0, 49: 0, 50: 0, 51:
0, 52: 0, 53: 1, 54: 1, 55: 0, 56: 0, 57: 1, 58: 0, 59: 1, 60: 0, 61: 0, 62: 1, 63: 0, 64: 0, 65: 0, 66: 0, 67: 1,
68: 0, 69: 0, 70: 0, 71: 0, 72: 0, 73: 0, 74: 0, 75: 0, 76: 0, 77: 0, 78: 0, 79: 1, 80: 0, 81: 0, 82: 0, 83: 0, 84:
0, 85: 1, 86: 0, 87: 0, 88: 0, 89: 1, 90: 0, 91: 0, 92: 0, 93: 0, 94: 0, 95: 0, 96: 0, 97: 0, 98: 0, 99: 1, 100: 0,
101: 0, 102: 0, 103: 0, 104: 0, 105: 0, 106: 0, 107: 0, 108: 0, 109: 0, 110: 0, 111: 0, 112: 0, 113: 0, 114: 0, 11
5: 0, 116: 0, 117: 0, 118: 0, 119: 0, 120: 0, 121: 0, 122: 0, 123: 0, 124: 1, 125: 0, 126: 0, 127: 0, 128: 0, 129:
0, 130: 0, 131: 0, 132: 0, 133: 0, 134: 1, 135: 0, 136: 0, 137: 1, 138: 0, 139: 0, 140: 0, 141: 0, 142: 0, 143: 0,
144: 0, 145: 0, 146: 0, 147: 0, 148: 0, 149: 0, 150: 0, 151: 0, 152: 1, 153: 0, 154: 0, 155: 0, 156: 0, 157: 0, 15
8: 0, 159: 0, 160: 0, 161: 0, 162: 1, 163: 0, 164: 0, 165: 0, 166: 0, 167: 1, 168: 0, 169: 0, 170: 0, 171: 0, 172:
0, 173: 0, 174: 0, 175: 0, 176: 0, 177: 0, 178: 1, 179: 0, 180: 0, 181: 0, 182: 0, 183: 0, 184: 1, 185: 0, 186: 0,
187: 0, 188: 0, 189: 0, 190: 0, 191: 1, 192: 0, 193: 0, 194: 1, 195: 1, 196: 1, 197: 0, 198: 0, 199: 0, 200: 1, 20
1: 0, 202: 0, 203: 0, 204: 0, 205: 0, 206: 0, 207: 0, 208: 0, 209: 0, 210: 0, 211: 0, 212: 1, 213: 0, 214: 0, 215:
0, 216: 1, 217: 0, 218: 0, 219: 1, 220: 0, 221: 0, 222: 0, 223: 0, 224: 0, 225: 0, 226: 0, 227: 0, 228: 0, 229: 0,
230: 0, 231: 1, 232: 0, 233: 0, 234: 0, 235: 0, 236: 0, 237: 0, 238: 1, 239: 0, 240: 0, 241: 0, 242: 0, 243: 0, 24
4: 0, 245: 0, 246: 0, 247: 0, 248: 1, 249: 0, 250: 0, 251: 0, 252: 0, 253: 0, 254: 0, 255: 0, 256: 0, 257: 1, 258:
1, 259: 1, 260: 1, 261: 0, 262: 0, 263: 0, 264: 0, 265: 0, 266: 0, 267: 0, 268: 0, 269: 1, 270: 1, 271: 0, 272: 0,
273: 1, 274: 0, 275: 0, 276: 1, 277: 0, 278: 0, 279: 0, 280: 0, 281: 0, 282: 0, 283: 0, 284: 0, 285: 0, 286: 0, 28
7: 0, 288: 0, 289: 0, 290: 0, 291: 1, 292: 1, 293: 0, 294: 0, 295: 0, 296: 0, 297: 0, 298: 1, 299: 0, 300: 1, 301:
0, 302: 0, 303: 0, 304: 1, 305: 0, 306: 1, 307: 1, 308: 1, 309: 0, 310: 1, 311: 1, 312: 1, 313: 1, 314: 0, 315: 0,
316: 0, 317: 1, 318: 0, 319: 1, 320: 1, 321: 0, 322: 0, 323: 1, 324: 1, 325: 0, 326: 1, 327: 0, 328: 1, 329: 0, 33
0: 1, 331: 0, 332: 0, 333: 0, 334: 0, 335: 1, 336: 0, 337: 0, 338: 1, 339: 0, 340: 0, 341: 1, 342: 1, 343: 0, 344:
0, 345: 0, 346: 1, 347: 1, 348: 0, 349: 0, 350: 0, 351: 0, 352: 0, 353: 0, 354: 0, 355: 0, 356: 0, 357: 1, 358: 1,
359: 0, 360: 0, 361: 0, 362: 0, 363: 0, 364: 0, 365: 0, 366: 0, 367: 1, 368: 0, 369: 0, 370: 1, 371: 0, 372: 0, 37
3: 0, 374: 0, 375: 0, 376: 1, 377: 0, 378: 0, 379: 0, 380: 0, 381: 1, 382: 0, 383: 0, 384: 1, 385: 0, 386: 0, 387:
0, 388: 1, 389: 0, 390: 1, 391: 0, 392: 0, 393: 0, 394: 1, 395: 0, 396: 0, 397: 0, 398: 0, 399: 0, 400: 1, 401: 0,
402: 0, 403: 0, 404: 0, 405: 0, 406: 0, 407: 0, 408: 1, 409: 0, 410: 0, 411: 0, 412: 0, 413: 1, 414: 0, 415: 0, 41
6: 0, 417: 1, 418: 1, 419: 0, 420: 0, 421: 0, 422: 0, 423: 0, 424: 0, 425: 0, 426: 0, 427: 1, 428: 1, 429: 0, 430:
0, 431: 0, 432: 0, 433: 1, 434: 0, 435: 0, 436: 1, 437: 0, 438: 1, 439: 0, 440: 0, 441: 1, 442: 0, 443: 0, 444: 1,
445: 0, 446: 1, 447: 1, 448: 0, 449: 0, 450: 0, 451: 0, 452: 0, 453: 0, 454: 0, 455: 0, 456: 0, 457: 0, 458: 1, 45
9: 1, 460: 0, 461: 0, 462: 0, 463: 0, 464: 0, 465: 0, 466: 0, 467: 0, 468: 0, 469: 0, 470: 0, 471: 0, 472: 0, 473:
1, 474: 1, 475: 0, 476: 0, 477: 0, 478: 0, 479: 0, 480: 0, 481: 0, 482: 0, 483: 0, 484: 0, 485: 0, 486: 0, 487: 1,
488: 0, 489: 0, 490: 0, 491: 0, 492: 0, 493: 0, 494: 0, 495: 0, 496: 0, 497: 1, 498: 0, 499: 1, 500: 0, 501: 0, 50
2: 0, 503: 0, 504: 0, 505: 1, 506: 0, 507: 1, 508: 0, 509: 0, 510: 0, 511: 0, 512: 0, 513: 0, 514: 1, 515: 0, 516:
0, 517: 1, 518: 0, 519: 1, 520: 0, 521: 1, 522: 0, 523: 0, 524: 1, 525: 0, 526: 0, 527: 1, 528: 0, 529: 0, 530: 0,
531: 1, 532: 0, 533: 0, 534: 0, 535: 0, 536: 1, 537: 0, 538: 1, 539: 0, 540: 1, 541: 1, 542: 0, 543: 0, 544: 0, 54
5: 0, 546: 0, 547: 1, 548: 0, 549: 0, 550: 1, 551: 0, 552: 0, 553: 0, 554: 0, 555: 0, 556: 0, 557: 1, 558: 0, 559:
1, 560: 0, 561: 0, 562: 0, 563: 0, 564: 0, 565: 0, 566: 0, 567: 0, 568: 0, 569: 0, 570: 0, 571: 0, 572: 1, 573: 0,
574: 0, 575: 0, 576: 0, 577: 1, 578: 1, 579: 0, 580: 0, 581: 1, 582: 1, 583: 0, 584: 0, 585: 0, 586: 1, 587: 0, 58
8: 0, 589: 0, 590: 0, 591: 0, 592: 1, 593: 0, 594: 0, 595: 0, 596: 0, 597: 1, 598: 0, 599: 0, 600: 0, 601: 1, 602:
0, 603: 0, 604: 0, 605: 0, 606: 0, 607: 0, 608: 0, 609: 1, 610: 1, 611: 0, 612: 0, 613: 0, 614: 0, 615: 0, 616: 1,
617: 0, 618: 0, 619: 1, 620: 0, 621: 0, 622: 0, 623: 0, 624: 0, 625: 0, 626: 0, 627: 0, 628: 1, 629: 0, 630: 0, 63
1: 0, 632: 0, 633: 0, 634: 0, 635: 0, 636: 1, 637: 0, 638: 0, 639: 0, 640: 0, 641: 0, 642: 1, 643: 0, 644: 0, 645:
0, 646: 0, 647: 0, 648: 0, 649: 0, 650: 0, 651: 0, 652: 1, 653: 0, 654: 0, 655: 0, 656: 0, 657: 0, 658: 0, 659: 0,
660: 0, 661: 0, 662: 0, 663: 0, 664: 0, 665: 0, 666: 0, 667: 0, 668: 0, 669: 0, 670: 1, 671: 1, 672: 0, 673: 0, 67
4: 0, 675: 0, 676: 0, 677: 0, 678: 0, 679: 0, 680: 0, 681: 0, 682: 0, 683: 0, 684: 0, 685: 0, 686: 0, 687: 0, 688:
0, 689: 0, 690: 1, 691: 0, 692: 0, 693: 0, 694: 0, 695: 0, 696: 0, 697: 0, 698: 0, 699: 0, 700: 0, 701: 1, 702: 0,
703: 0, 704: 0, 705: 0, 706: 0, 707: 1, 708: 0, 709: 1, 710: 0, 711: 1, 712: 0, 713: 0, 714: 0, 715: 0, 716: 0, 71
7: 1, 718: 1, 719: 0, 720: 0, 721: 1, 722: 0, 723: 0, 724: 0, 725: 0, 726: 0, 727: 1, 728: 0, 729: 0, 730: 0, 731:
1, 732: 0, 733: 0, 734: 0, 735: 0, 736: 0, 737: 0, 738: 0, 739: 0, 740: 0, 741: 0, 742: 0, 743: 1, 744: 0, 745: 0,
746: 0, 747: 0, 748: 1, 749: 0, 750: 0, 751: 1, 752: 0, 753: 0, 754: 0, 755: 1, 756: 1, 757: 0, 758: 0, 759: 0, 76
0: 1, 761: 0, 762: 0, 763: 0, 764: 1, 765: 0, 766: 1, 767: 0, 768: 0, 769: 0, 770: 0, 771: 0, 772: 0, 773: 1, 774:
0, 775: 1, 776: 0, 777: 0, 778: 0, 779: 0, 780: 1, 781: 0, 782: 1, 783: 0, 784: 0, 785: 0, 786: 0, 787: 0, 788: 0,
789: 0, 790: 0, 791: 0, 792: 0, 793: 0, 794: 0, 795: 0, 796: 0, 797: 1, 798: 0, 799: 0, 800: 0, 801: 0, 802: 1, 80
3: 1, 804: 0, 805: 0, 806: 0, 807: 0, 808: 0, 809: 0, 810: 1, 811: 0, 812: 0, 813: 0, 814: 0, 815: 0, 816: 0, 817:
0, 818: 0, 819: 0, 820: 0, 821: 1, 822: 0, 823: 0, 824: 0, 825: 0, 826: 0, 827: 0, 828: 1, 829: 0, 830: 1, 831: 0,
832: 1, 833: 0, 834: 0, 835: 0, 836: 1, 837: 0, 838: 0, 839: 0, 840: 0, 841: 0, 842: 0, 843: 1, 844: 0, 845: 0, 84
6: 0, 847: 0, 848: 0, 849: 0, 850: 1, 851: 0, 852: 0, 853: 0, 854: 1, 855: 1, 856: 0, 857: 1, 858: 0, 859: 0, 860:
0, 861: 0, 862: 0, 863: 1, 864: 0, 865: 0, 866: 1, 867: 1, 868: 0, 869: 0, 870: 0, 871: 0, 872: 1, 873: 0, 874: 0,
875: 1, 876: 0, 877: 0, 878: 0, 879: 0, 880: 1, 881: 1, 882: 0, 883: 0, 884: 0, 885: 0, 886: 0, 887: 0, 888: 1, 88
9: 0, 890: 0, 891: 0}
```

In [23]:

```
import pandas

def add_full_name(path_to_csv, path_to_new_csv):
    df=pandas.read_csv(path_to_csv)
#   for Hank Aaron, nameFull would be 'Hank Aaron',
    df['nameLast']=df['nameFirst'].map(str)+' '+df['nameLast'].map(str)
    #2) The data in the pandas dataFrame to a new csv file located at path_to_new_csv
    df.to_csv(path_to_new_csv)

if __name__ == "__main__":
    # For local use only
    # If you are running this on your own machine add the path to the
    # Lahman baseball csv and a path for the new csv.
    # The dataset can be downloaded from this website: http://www.seanlahman.com/baseball-archive/statistics
    # We are using the file Master/people.csv
    path_to_csv = r"People.csv"
    path_to_new_csv = r"New data.csv"
    add_full_name(path_to_csv,path_to_new_csv)
    names=pandas.read_csv(path_to_new_csv)
    print(names['nameLast'])
```

```
0          David Aardsma
1             Hank Aaron
2           Tommie Aaron
3               Don Aase
4              Andy Abad
                 ...
20365         Frank Zupo
20366       Paul Zuvella
20367    George Zuverink
20368     Dutch Zwilling
20369          Tony Zych
Name: nameLast, Length: 20370, dtype: object
```

In [26]:

```
import pandas
import pandasql

def select_first_50(filename):
    # Read in our aadhaar_data csv to a pandas dataframe.  Afterwards, we rename the columns
    aadhaar_data = pandas.read_csv(filename)
    aadhaar_data.rename(columns = lambda x: x.replace(' ', '_').lower(), inplace=True)
#     print(aadhaar_data)
    # Select out the first 50 values for "registrar" and "enrolment_agency"
    # in the aadhaar_data table using SQL syntax.
    #
    # Note that "enrolment_agency" is spelled with one l. Also, the order
    # of the select does matter. Make sure you select registrar then enrolment agency
    # in your query.
    #
    # You can download a copy of the aadhaar data that we are passing
    # into this exercise below:
    # https://s3.amazonaws.com/content.udacity-data.com/courses/ud359/aadhaar_data.csv
    q = """
    SELECT registrar,enrolment_agency FROM aadhaar_data LIMIT 50;
    """
    #Execute your SQL command against the pandas frame
    aadhaar_solution = pandasql.sqldf(q.lower(), locals())
    return aadhaar_solution
if __name__ == "__main__":
    rows=select_first_50(r"aadhaar_data.csv")
    print(rows)
```

```
        registrar             enrolment_agency
0   Allahabad Bank            Tera Software Ltd
1   Allahabad Bank            Tera Software Ltd
2   Allahabad Bank  Vakrangee Softwares Limited
3   Allahabad Bank  Vakrangee Softwares Limited
4   Allahabad Bank  Vakrangee Softwares Limited
5   Allahabad Bank  Vakrangee Softwares Limited
6   Allahabad Bank  Vakrangee Softwares Limited
7   Allahabad Bank  Vakrangee Softwares Limited
8   Allahabad Bank  Vakrangee Softwares Limited
9   Allahabad Bank  Vakrangee Softwares Limited
10  Allahabad Bank  Vakrangee Softwares Limited
11  Allahabad Bank  Vakrangee Softwares Limited
12  Allahabad Bank  Vakrangee Softwares Limited
13  Allahabad Bank  Vakrangee Softwares Limited
14  Allahabad Bank  Vakrangee Softwares Limited
15  Allahabad Bank  Vakrangee Softwares Limited
16  Allahabad Bank  Vakrangee Softwares Limited
17  Allahabad Bank  Vakrangee Softwares Limited
18  Allahabad Bank  Vakrangee Softwares Limited
19  Allahabad Bank  Vakrangee Softwares Limited
20  Allahabad Bank  Vakrangee Softwares Limited
21  Allahabad Bank  Vakrangee Softwares Limited
22  Allahabad Bank  Vakrangee Softwares Limited
23  Allahabad Bank  Vakrangee Softwares Limited
24  Allahabad Bank  Vakrangee Softwares Limited
25  Allahabad Bank  Vakrangee Softwares Limited
26  Allahabad Bank  Vakrangee Softwares Limited
27  Allahabad Bank  Vakrangee Softwares Limited
28  Allahabad Bank  Vakrangee Softwares Limited
29  Allahabad Bank  Vakrangee Softwares Limited
30  Allahabad Bank  Vakrangee Softwares Limited
31  Allahabad Bank  Vakrangee Softwares Limited
32  Allahabad Bank  Vakrangee Softwares Limited
33  Allahabad Bank  Vakrangee Softwares Limited
34  Allahabad Bank  Vakrangee Softwares Limited
35  Allahabad Bank  Vakrangee Softwares Limited
36  Allahabad Bank  Vakrangee Softwares Limited
37  Allahabad Bank  Vakrangee Softwares Limited
38  Allahabad Bank  Vakrangee Softwares Limited
39  Allahabad Bank  Vakrangee Softwares Limited
40  Allahabad Bank  Vakrangee Softwares Limited
41  Allahabad Bank  Vakrangee Softwares Limited
42  Allahabad Bank  Vakrangee Softwares Limited
43  Allahabad Bank  Vakrangee Softwares Limited
44  Allahabad Bank  Vakrangee Softwares Limited
45  Allahabad Bank  Vakrangee Softwares Limited
46  Allahabad Bank  Vakrangee Softwares Limited
47  Allahabad Bank  Vakrangee Softwares Limited
48  Allahabad Bank  Vakrangee Softwares Limited
49  Allahabad Bank  Vakrangee Softwares Limited
```

In [4]:

```
import json
import requests
import pprint
def api_get_request(url):
    # In this exercise, you want to call the last.fm API to get a list of the
    # top artists in Spain. The grader will supply the URL as an argument to
    # the function; assume get method returns json data with key “top_10”
    # you do not need to construct the address or call this
    # function in your grader submission.
    #
    # Once you've done this, return the name of the number 1 top artist in
    # Spain.
    data = requests.get(url).text
    data = json.loads(data)

    pp = pprint.PrettyPrinter(indent=2)
    pp.pprint(data['topartists']['artist'][0]['name'])

    return data['topartists']['artist'][0]['name']
if __name__ == "__main__":
    url = 'http://ws.audioscrobbler.com/2.0/?method=geo.gettopartists&country=spain&api_key=4beab33cc6d65b05800d51f5e83bde1b&form
    print(api_get_request(url))
```

```
'David Bowie'
David Bowie
```

In [5]:

```
import pandas as pd
import numpy as np


def imputation(filename):
    # Pandas dataframes have a method called 'fillna(value)', such that you can
    # pass in a single value to replace any NAs in a dataframe or series. You
    # can call it like this:
    #     dataframe['column'] = dataframe['column'].fillna(value)
    #
    # Using the numpy.mean function, which calculates the mean of a numpy
    # array, impute any missing values in our Lahman baseball
    # data sets 'weight' column by setting them equal to the average weight.
    #
    # You can access the 'weight' colum in the baseball data frame by
    # calling baseball['weight']

    baseball = pd.read_csv(filename)
    baseball['weight'].fillna(baseball['weight'].mean(), inplace=True)

    return baseball
if __name__ == "__main__":
    filename = 'People.csv'
    output=imputation(filename)
    print(output['weight'])
```

```
0        215.0
1        180.0
2        190.0
3        190.0
4        184.0
         ...
20365    182.0
20366    173.0
20367    195.0
20368    160.0
20369    190.0
Name: weight, Length: 20370, dtype: float64
```

In [ ]:

In [ ]: