Generating random identities with a python program

Abstract

Generating random identities can be useful in lots of situations, especially when you need them to test a database or something similar. Therefore we wrote this program to generate some identites with their name and address.

Table of Contents

Abstract....1

1. The statistics....2

2. The given tasks....3

    2.1 Simple name generation....4

    2.2 Generating full names....5

    2.3 Statistical test 1....7

    2.4 Full identity generation....8

    2.5 Full statistical test and sorted list....9

        2.5.1 Sorted list....9

        2.5.2 Statistical test....9

3. User Interaction....10

4. Usage....11

    4.1 Using the program....11

    4.2 Adding more names....11

5. Thanks for reading....12

Oliver Theobald, 7146127 EPR16 Tristan Stadler, 6956404

1. The statistics

There are lots of statistics about the distribution of names. Some were given by the sheet, some are self-made. Here are the ones that we used for our program.

I. The distribution between males and females is about equal.

II. 10% of all persons have a double first-name (like „Franz-Josef“) which is always separated by a „-“.

III. 15% of all persons have a double last-name (like „Meier-Müller“) which is always separated by a „-“.

IV. Double names can‘t consist of two similar names (like „Meier-Meier“). This goes for both first and last names.

V. 1% of all persons have a PhD.

VI. Of those who have a PhD, 45% are women and 55% are men. (Source: See exercise sheet).

VII. Of all house numbers, 1% have 4 digits, 10% have 3 digits and the remaining 89% have 1 or 2 digits. This rule is self-made and based on the assumption that in Germany the higher the number is the more uncommon it gets.

VIII. The distribution of the prefix and suffix of the streetname is:

Haupt-	10%	...Straße	78%
Schul-	8%	...Weg	18%
Garten-	7%	...Allee	2%
Dorf-	7%	...Ring	1%
Bahnhof-	7%	...Platz	1%
Wiesen-	7%		
Berg-	6%		
Kirch-	4%		
Wald-	4%		
Ring-	4%		
Other*	36%		

* Other prefixes include names of flowers, trees, cities and celebrities.

And that‘s all. That‘s every statistic that we use.

One last thing: We were given a list with male and female first names as well as some last names. We changed the type of them from sets to lists, but content-wise we didn‘t change anything within those lists. We renamed it from „EPR_04.py“ to „names.py“ since that fits the purpose better.

2. The given tasks

We were given a number of tasks to complete. They each base upon the previous tasks and are designed to get a full and functional program at the end while always having an idea on what to do next.

2.1 Simple name generation

The first task was to simply generate single first and last names (no double names yet). Here is an example:

Male first names:	Female first names:	Last names:
Ringo	Hella	Schillung
Hans	Augusta	Schinder
Hugo	Cemile	Schillung
Fabius	Adina	Herzog
Antonius	Heidi	Siemens
Johann	Sascha	Andres
Tristan	Therese	Landzettel
Silvester	Xanthippe	Abraham
Siegfried	Emma	Herrmann
Aldhelm	Leyla	Neumann
Harro	Diethild	Schmid
Silvio	Bettina	Wieland
Elias	Delia	Schulze
Jürgen	Georgia	Hering
Bert	Waltraud	Calvin
Adam	Jo	Schulz
Philemon	Marietta	Noack
Joah	Antoinette	Zacharias
Micha	Merrit	Winkler
Hubert	Mandy	Träger
Harro	Magdalena	Preuß
Thorbjörn	Nefertari	Reich
Vivien	Judith	Mach
Ewald	Synke	Landzettel
Paul	Fabia	Sauerbruch
Burckhardt	Sabrina	Schulze
Vivien	Solveig	Götz
Leonhard	Samantha	Huber
Wieland	Antje	Thomas
Nikolai	Antoinette	Schnorr
Olaf	Else	Richter
Patrick	Anka	Kästner
Yannick	Alwine	Schröder
Béla	Lea	Oswald
Marvin	Vera	Ineichen
Artur	Esther	Thoma
Tristan	Claudia	Wittgenstein
Conan	Augustine	Kaiser
Bertram	Waltraud	Frenzel
Thorsten	Jelena	Franz
Mohammed	Rike	Rohlfs
Modestus	Anneliese	Thode
Bartholom	Jeannette	Frenzel
Philipp	Helene	Krämer
Sebastian	Winifred	Fröhlich
Bonifatius	Genoveva	Schleich
Chlothar	Helgard	Sommerfeld
Andrea	Jana	Siemens
Hermann	Gundela	Dürrschnabel
Chlothar	Angelina	Hoffmann

In the console they are all written beneath each other. We‘ve put them in a table here to increase readability and reduce the length of this document.

These are done by repeatedly calling the function male_name(), female_name() and last_name() respectively.

2.2 Generating full names

The next step is to generate full names (this includes the PhD title). We have to make sure that requirements I to VI are met. This is done by the function full_name(). But how?

I. An equal distribution of both male and female names is reached by generating a random integer between 1 and 100. If the value is within the range of 1 to 50, the generated person will be male, if the value is within 51 to 100, the person will be female. Therefore the distribution should be about 50/50.

II. Another random number between 1 and 100. If the value is below or equal to 10, the person will have a double first name, else it will be a single first name. So there is a 10% chance of having a double first name which is the requirement.

III. Another random number between 1 and 100. If the value is between 40 and 54 (which are 15 possible values) the person will have a double last name. Why 40 and 54? Why not. It doesn‘t really matter, it‘s just to have some other numbers.

IV. This is quite easy. When generating a double name, the function double_name(gender) is called, where the „gender“ variable tells the function if the double name has to be a last, a male first or a female first name. Then it calls either the function double_name_last(), double_name_male() or double_name_female(), where each function generates random names until the names are different. When they are different, they are connected by a „-“ and then returned to double_name(gender), which returns the double name to full_name().

V. Simply another random number. This can only be done together with VI, therefore it will be explained there.

VI. We got this by using Bayes‘ theorem. We generate a random number between 1 and 1,000. If the number is less or equal to 11 and the person is male, it will get a PhD. If the number is less or equal to 9 and the person is female, it will get a PhD as well. Why this works? Let‘s take a look at the values. I won‘t proof Bayes‘ theorem here, however I will show that the results we got give us the correct result.

    50% or 0.5 out of 1 are male and 50% or 0.5 out of 1 are female. When we generate a random number between 1 and 1000, it has a 11/1000 chance to be less or equal to 11, which is equal to 0.011 or 1.1%. The chance of the number being less or equal to 9 is therefore 0.009 or 0.9%. Now we calculate: 0.5 (for being male) * 0.011 (for having a PhD as male) + 0.5 (for being female) * 0.009 (for having a PhD as a female) = 0.0055 + 0.0045 = 0.01 = 1%. Therefore the chance of having a PhD is 1% overall, whereas 55% of the graduates are male and 45% are female.

Now that we have the math out of the way, we can look at some example values. Here we have 100 full names.

Irene Haller	Daniela Albrecht	Antonius Amalie-Resch
Herr Sascha-Igor Albert	Lisanne Sygnetzki	Meike Schubert
Cornelia Hoffmann	Cosima Forge	Gertrud Schnur
Susanne-Asta Krause	Baldur Schröder	Malwine Abt-Albrecht
Melanie Widera	Samuel Haller	Elton Lehner
Maike Friedemann	Béla Nordmann	Veronika Frank
Arved Schneider	Gundo Hoffmann-Freud	Melanie Richter
Leyla Körner	Melanie Henne	Brigitte Anders
Oliver-Simon Weber	Lilli Löchte	Radomir Sonntag
William Amalie	Charles Götz	Aldhelm-Andrea Thode
Birger Schulthess	Georgia Schinder	Beate-Holde Plagemann
Sean Hünlein	Alwin Lai	Justus Falk-Schnur
Bernhard Grubner-Oswald	Gundela Kosba	Rainer Ingo
Jürgen Friedemann	Hauke Honecker	Jördis Schmitt
Margarete Calvin	Louis Resch	Lilli Kurz
Aaron Becherer	Christiane Leber	Eudokia-Kira Zahn
Brigitte-Hannelore Krause	Silas Sam	Oswald Haller-Thoma
Irma Rohlfs	Sophia Hünlein	Paul Christiansen-Abt
Berit Friedemann-Kosba	Sönke Körner	Rudolf-Joah Jauch
Wolfhard Resch	Beatrix Micus	Hilde Frenzel
Klemens Knebel	Nico Hintz	Volker Zacharias-Schmid
Balduin Huber-Landzettel	Edeltraud Franz	Edeltraud Micus
Samantha-Heilgard Ineichen	Manuela Götz	Adele Voigtländer
Aloisia Sauerbruch-Schmidt	Crispian Schulz	Peppone Landzettel-Gärtner
Fennedine Träger	Arndt Kaiser	Johannes Kolmar-Semper
Frau Nikola-Solvej Tannhäuser	Roald Franz-Querfurt	Michaela Ines-Ruoff
Lydia Ines-Haller	Dr. Zenon Freund	Timon Eschenburg
Janna-Meral Peter	Aribert Amalie	Lukas Vogt
Ruprecht Zacharias	Denise Zacharias	Heilgard-Anke Ludwig
Hartwin Oswald-Schröder	Janine Meier	Christine Frenzel
Robert Jakobi	Ingo Reich	Luca Ingo
Anja-Anabel Winkler	Simon-Peppone Wenzel-Meier	Geoffrey Noack
Sebastian Sander-Schnurre	Bert-Nico Sygnetzki	Corinna Nietzsche-Kosba
Waldemar Jauch		

As you can see, there is 1 PhD out of 100 people. There are some double first names and double last names as well.

Note: We added the prefix „Herr“ or „Frau“ for male and female names respectively in case that the first name can be used for both genders. It‘s simply to identify the gender of a person.

Sascha-Igor Albert is a great example. If you have a person named „Sascha“ it could be both male and female. The second part of the double name is useless. Why? Do you know Christoph-Maria Herbst, also known as Stromberg? Maria is a female name but can be used as second part of a double name by male persons as well. Therefore we simply add „Herr“ or „Frau“ to make it easier.

2.3 Statistical test 1

Just implementing the requirements is not enough, we need to test our implementation. Therefore the function statistical_test_name(sample_size) creates random names (in this case 1,000) and tests them on the statistics. 1% doctors, 10% with double first name, 15% with double last name, 50% male and 50% female.

An example result:

```
0.8% are doctors.
9.7% have a double first name.
16.1% have a double last name.
49.1% are male.
50.9% are female.
```

So why are the numbers slightly off? Because of the law of large numbers which essentially says: „The larger the sample size, the better the statistical result.“ 1,000 is simply not a large enough sample to get accurate values. But since all of them are near of what they should be, it is fine.

Note: Theoretically you should have printed the 1,000 used names as well, but I won‘t include them here for the sake of keeping the document below 50 pages.

2.4 Full identity generation

The next task is to implement the address generation (done by address()) and combining them with the names to get full identites (done by identity()). The percentage values are implemented exactly like before, by generating a random number between 1 and 100 and then setting the range for each option according to the value. This is used for the prefix (if „other“ is the result, the function will take a random element of the set in names.py), the suffix and the house number (here it will adjust the range in which the house number will be randomly generated.

Muriel Herold, Kiefernstraße 14
Dennis Nietzsche, Ringstraße 979
Fridolin-Klemens Kühn, Waldstraße 19
Tina-Vera Kühn, Bahnhofstraße 84
Geofreda Löchte, Bergstraße 53
Ilona Reich, Kirchstraße 38
Quintus Schulz, Fichtenstraße 33
Frauke Sauerbruch, Hauptstraße 45
Eudokia Falk, Dorfweg 10
Keiko Sam, Hauptstraße 24
Hilde Türk, Hauptweg 85
Patrick Steiner, Edelweißstraße 9
Gunther Köhler, Ulmenstraße 17
Jörn Semper, Hauptstraße 23
Kurt Ines, Gartenstraße 557
Dr. Eduard Oswald, Lilienallee 56
Jesus Reich-Eschenburg, Gartenweg 49
Josef-Julier Siemens, Buchenstraße 142
Delia-Helgard Schultz, Rosenstraße 760
Pascal Jörg, Gartenstraße 25
Olga Zahn, Hauptstraße 8
Ingmar Winkler, Eichenstraße 24
Malwine Plagemann, Wiesenring 82
Archibald Widera, Waldstraße 70
Carolin Amalie, Hauptstraße 91
Florin Schulthess-Gruber, Hauptstraße 61
Andrzej Böhm, Rotdornallee 35
Josephine Wieland, Blauregenstraße 17
Hilde Pohl, Waldstraße 81
Eudokia Sonntag, Pilsener Straße 86
Adrian Wurm-Semper, Wiesenstraße 15
Corinna-Veronika Krause-Siemens, Tulpenstraße 318
Agnes-Olga Gerster, Hauptstraße 52
Armin Rumpf, Schwarzeichenstraße 85
Calvin Schulze-Eschenburg, Schulstraße 60
Calvin Türk, Hyazinthenstraße 12
Torben-Benno Lücking, Wiesenstraße 47
Irmgard Schmidt, Ringstraße 372
Weselius-Edgar Kästner, Kirchstraße 4234
Annegret Türck, Schulstraße 73
Karoline Böll, Eichenstraße 92
Dominik-Inga Tenberge, Wiesenweg 8
Adelgunde Rumpf-Kühn, Dorfstraße 62
Clementine Abendroth, Frankfurter Straße 419
Genoveva Thyssen, Waldstraße 8698
Salomo Böll, Wiesenstraße 197
Frau Kari Ossege-Jauch, Ulmenstraße 55
Grant Nordmann, Hauptstraße 372
Elias Böll, Gartenstraße 120
Nikolai Overbeck, Dorfstraße 64
Thorbjörn Thomas, Hauptstraße 61
Frau Ira Nordmann, Stechpalmenstraße 67
Silas Herrmann, Hauptstraße 72
Tim Neumann, Hauptstraße 644
Veronika Wittgenstein, Tulpenstraße 95
Aribert Krause, Bergstraße 68
Othmar Krüger, Schulstraße 24
Rupert Thoma, Quackenbrücker Straße 43
Kriemhild Krupp, Hauptstraße 63
Petrus Abt, Saarbrücker Weg 81
Gerlinde Pügner, Bergstraße 73
Justus Nordmann, Berglingstraße 50
Harro Ludwig, Schulstraße 115
Eudokia Tannhäuser, Fichtenstraße 52
Boris Jörg, Schulweg 393
Radomir Mester, Ringweg 45
Dr. Xaver Jung, Hauptweg 39
Almut Becherer, Schulstraße 62
Otto Weck-Herzog, Leninweg 77
Veronika Winkler-Schulze, Medicistraße 48
Adelinde Howaldt, Dorfstraße 38
Götz Wieland, Rosenstraße 63
Dr. Toni Franz, Bergstraße 320
Lilith Schillung, Bahnhofstraße 24
Viola Baumann-Schinder, Bahnhofstraße 80
Fabius Weck, Dorfstraße 11
Andre Behm, Hauptstraße 52
Crispian Selbmann, Bahnhofweg 14
Anastasia Schnurrer, Dorfstraße 49
Sabine Sonntag, Schulstraße 69
Leo Mach, Guevaraweg 11
Veronika-Charlotte Luhmann, Gartenstraße 29
Mike Becker, Schneeglöckchenstraße 17
Friedemann Frank, Wiesenweg 49
Wieland-Boleslaw Amalie, Rotbuchenstraße 24
Reinhard Schnurrer, Hegelstraße 77
Fred Abraham, Hauptstraße 16
Philipp Braun, Dorfstraße 16
Julier Winkler, Wiesenallee 24
Hubert Frenzel, Clematisstraße 406
Christopher-Eckbert Lehmann, Dorfstraße 70
Emma Nordmann, Ringweg 93
Arnold Grubner, Dorfweg 45
Maike Krause, Zedernstraße 31
Heinrich Steiner, Leipziger Weg 11
Wladyslaw Krause, Bergweg 89
Sarine Wittgenstein, Schneerosenweg 61
Arno Schneider, Waldstraße 2
Asmus Körner, Dorfstraße 83
Hartmut Quant, Hauptstraße 85

Again, In the program all names are printed in a single list and the columns here are just to shorten the text a bit.

2.5 Full statistical test and sorted list

We need to generate a sorted list of persons and do a full statistical analysis.

2.5.1 Sorted list

The solution for the sorted list is inspired by this stackoverflow post: https://stackoverflow.com/questions/20459982/how-to-sort-a-list-alphabetically-by-the-second-word-in-a-string

Essentially we use two things: The function sorted() and a lambda expression.

Sorted simply returns a sorted list. We use the lambda expression as key. The key simply defines by what to sort, e.g. you could sort by the length of the words etc.

lambda simply creates small functions that are restricted to a single expression. This allows us to create our small and simple function where it is required instead of writing a whole new function with docstring and return statement and so on. In this case our lambda is used to sort the sample list first by their first names, then by their last names. More on lambda here: https://docs.python.org/2/tutorial/controlflow.html#lambda-expressions

And that‘s all. Since Pythons sorting is stable (repeated elements are sorted as they appear in the list), we can simply sort the sample by the first name of the persons and then by the last name, therefore keeping the sorting by the first name where it is necessary (people with the same last name).

2.5.2 Statistical test

Not much to say here, it is essentially like the first statistical test (2.3) just with some more variables. The testing of the name is the same, it even used the already written function. The testing of the address is done by test_address(residence) where the function returns you a list with 19 elements, each representing one part (0-10 are prefixes, 11-15 suffixes, 16-18 house number length). The values that apply to the given address are set to 1, the others remain 0. After that everything is added to the counter for the whole sample. Rinse and repeat until it‘s done for every address in the sample. From there you can simply calculate the percentage.

3. User Interaction

To make the program easy to use, we implemented a user interaction. Every possibility will be asked:

- Generating single male names
- Generating single female names
- Generating single last names
- Generating double male names
- Generating double female names
- Generating double last names
- Generating full names
- Statistically test names
- Generating full identities
- Generating a sorted list of names
- Creating a full statistical test.

The order of these is based on their order in the given tasks and therefore their time of implementation.

4. Usage

What you can do are – simply put – two things:

1. You can use the program as it is or
2. you can add more names in names.py – for persons and/or streets.

4.1 Using the program

This is really simple. By running the program as main (or alternatively calling the user_interaction() function) you will get asked a couple of questions (see 3.).

If your answer to a question is yes, don‘t enter „yes“!!! Just enter the number of elements you want to be created (for the statistical tests this will be the sample size, for the others it‘s just the number of elements you get) as an integer.

If you don‘t want to do it, just enter anything else. Can be a „no“, can be a „5234.1234,531426,321,“ or a „f*ck off“. Just anything that isn‘t an integer.

4.2 Adding more names

The names contained in names.py are just an example, feel free to add and remove whatever you like. However, a few notes:

- There is no checking whether the name is already in the list you want to add it to. However, the lists referring to person names are currently alphabetically sorted.
- Street names are generated by putting a prefix and a suffix together. Therefore the prefix should be capitalized.
- If using a flower or a tree (in german), use the plural. Simple check: What sounds better? „Tannenstraße“ or „Tannestraße“.
- Celebrities are normally referred to by their last names.
- Be careful with city names. Normally you won‘t find anything like a „Frankfurtstraße“, more like a „Frankfurter Straße“. If you want to use a city name: Write „{city name}er „ instead of „{city name}“. By adding a „er“ and a blank directly after your city of choice, you will generate the opportunity of creating good sounding names instead of names that suck.

5. Thanks for reading

Thank you for reading this documentation. I hope you like our program. Feel free to contact us if you have questions or find errors/mistakes.