impracticalpythonprojects

Release 0.7.0

Jose A. Lerma III

May 20, 2019

MODULE REFERENCE

Example implementations of the practice and challenge projects in Impractical Python Projects.

It's a fantastic intermediate level book that has truly impractical (but fun) projects. It's a great way to get tricked into learning new conventions, techniques, and modules.

My original python-tutorials repository is already very nested, so these will be easier to find and review here; however, the original repository still has relevant information about configuring a Python environment/IDE.

Bonus content includes Google style docstrings (such wow), main functions (so standard), pip requirements files (so helpful), and test files (**not** punny at all).

CHAPTER

ONE

SRC

1.1 src package

1.1.1 Subpackages

src.ch01 package

Subpackages

src.ch01.challenge package

Submodules

src.ch01.challenge.c1_foreign_bar_chart module

Return letter 'bar chart' of a non-English sentence.

src.ch01.challenge.c1_foreign_bar_chart.**add_keys_to_dict**(*dictionary: dict*) → dict
Add keys to dictionary.

Check keys of a letter dictionary and add missing letters.

Parameters **dictionary** (*dict*) – Dictionary to check keys of.

Returns Dictionary with all ASCII lowercase letters as keys.

Raises **TypeError** – If *dictionary* is not a dictionary.

src.ch01.challenge.c1_foreign_bar_chart.**foreign_freq_analysis**(*sentence: str*) → dict
Wrap freq_analysis and add_keys_to_dict.

Passes given sentence through freq_analysis then add_keys_to_dict to fill in missing keys.

Parameters **sentence** (*str*) – String to count letters of.

Returns Dictionary with all ASCII lowercase letters as keys and list with letters repeated based on their frequency as values.

src.ch01.challenge.c1_foreign_bar_chart.**main**()
Demonstrates the Foreign Bar Chart.

src.ch01.challenge.c2_name_generator module

Generate random names from a list of names.

src.ch01.challenge.c2_name_generator.**add_name_to_key**(*name: str*, *dictionary: dict*, *key: str*) → None

Add name to key in dictionary.

Add name to dictionary under key if not already present.

Parameters

- **name** (*str*) – Name to add to dictionary.
- **key** (*str*) – Key to add name under.
- **dictionary** (*dict*) – Dictionary to add name to.

Returns None. Name is added under key if not present, dictionary is unchanged otherwise.

Raises

- **TypeError** – If name and key aren't strings or if dictionary isn't a
- **dictionary.** –

src.ch01.challenge.c2_name_generator.**build_name_list**(*folderpath: str*) → list

Build name list from folder.

Builds list of names from name files in given folder.

Parameters **folderpath** (*str*) – Path to folder with name files.

Returns List with names from folderpath.

Raises **IndexError** – If folderpath has no `.txt` files.

src.ch01.challenge.c2_name_generator.**generate_name**(*name_dict: dict*) → str

Generate pseudo-random name.

Use names in dictionary to generate a random name.

Parameters **name_dict** – Dictionary from split_names.

Returns String with a random name.

Raises **KeyError** – If there aren't three keys in the dictionary.

Note: Only add middle name between 1/3 and 1/4 of the time.

---

src.ch01.challenge.c2_name_generator.**main**()

Demonstrate name generator.

src.ch01.challenge.c2_name_generator.**name_generator**(*folderpath: str*) → str

Wrap generate_name, split_names, and build_name_list.

Passes given folderpath through build_name_list to get the names in a list, then split_names to split them into a dictionary, and finally through generate_name to make the actual name.

Parameters **folderpath** (*str*) – Path to folder with name files.

Returns String with random name.

src.ch01.challenge.c2_name_generator.**read_from_file**(*filepath: str*) → list
Read from file.

Reads lines from text file and returns a list.

Parameters **filepath** (*str*) – Path to file with names.

Returns List with each line from the file as an element.

Note: Removes trailing whitespaces.

src.ch01.challenge.c2_name_generator.**split_names**(*name_list: list*) → dict
Split names from list of names.

Splits first, middle, and last names from a given list of names.

Parameters **name_list** (*list*) – List with names as elements.

Returns Dictionary of lists with `first`, `middle`, and `last` as keys and names as values.

Raises

- **TypeError** – If given name list is not a list or tuple.
- **ValueError** – If given name list is empty.

Note: Drops suffix and adds nickname to middle names.

Module contents

Chapter 1 Challenge Projects.

src.ch01.challenge.**ADD_KEYS_ERROR**
String with TypeError for Foreign Bar Chart.

Type str

src.ch01.challenge.**SPLIT_NAME_LIST_ERROR**
String with TypeError for Name Generator.

Type str

src.ch01.challenge.**SPLIT_NAME_EMPTY_ERROR**
Sting with ValueError for Name Generator.

Type str

src.ch01.challenge.**ADD_NAME_TO_KEY_ERROR**
String with TypeError for Name Generator.

Type str

src.ch01.challenge.**GENERATE_NAME_ERROR**
String with KeyError for Name Generator.

Type str

src.ch01.challenge.**BUILD_LIST_ERROR**
String with IndexError for Name Generator.

Type str

src.ch01.practice package

Submodules

src.ch01.practice.p1_pig_latin module

Takes a word as input and returns its Pig Latin equivalent.

src.ch01.practice.p1_pig_latin.**encode**(*word: str*) → str
Check if word starts with vowel, then translate to Pig Latin.

If a word begins with a consonant, move the consonant to the end of the word and add 'ay' to the end of the new word. If a word begins with a vowel, add 'way' to the end of the word.

Parameters **word** (*str*) – Word to encode to Pig Latin.

Returns Encoded Pig Latin word.

Raises **TypeError** – If *word* is not a string.

src.ch01.practice.p1_pig_latin.**main**()
Demonstrate Pig Latin encoder.

src.ch01.practice.p2_poor_bar_chart module

Takes a sentence as input and returns a 'bar chart' of each letter.

src.ch01.practice.p2_poor_bar_chart.**freq_analysis**(*sentence: str*) → dict
Perform frequency analysis of letters in sentence.

Iterate through each letter in the sentence and add it to a dictionary of lists using `collections.defaultdict`.

Parameters **sentence** (*str*) – String to count letters of.

Returns Dictionary with each letter as keys and list with letters repeated based on their frequency as values.

Example

```
>>> from src.ch01.practice.p2_poor_bar_chart import freq_analysis
>>> test = 'aaabbbccc'
>>> freq_analysis(test)
defaultdict(<class 'list'>, {'a': ['a', 'a', 'a'],
                             'b': ['b', 'b', 'b'],
                             'c': ['c', 'c', 'c']})
```

Raises **TypeError** – If *sentence* is not a string.

src.ch01.practice.p2_poor_bar_chart.**main**()
Demonstrates the Poor Bar Chart.

src.ch01.practice.p2_poor_bar_chart.**print_bar_chart**(*freq_dict: dict*) → None
Print dictionary to terminal.

Use `pprint` to print dictionary with letter frequency analysis to terminal.

Parameters

- **freq_dict** (*dict*) – Dictionary with frequency analysis from
- **freq_analysis.** –

Returns If recursive, prints a recursive-safe string, otherwise prints the dictionary.

Raises **TypeError** – If *freq_dict* is not a dictionary.

Module contents

Chapter 1 Practice Projects.

src.ch01.practice.**VOWELS**
Tuple containing characters of the English vowels (except for 'y')

Type tuple

src.ch01.practice.**ENCODE_ERROR**
String with TypeError for Pig Latin Encoder

Type str

src.ch01.practice.**FREQ_ANALYSIS_ERROR**
String with TypeError for Poor Bar Chart frequency analysis

Type str

src.ch01.practice.**PRINT_BAR_CHART_ERROR**
String with TypeError for Poor Bar Chart printer.

Type str

Module contents

Chapter 1.

1.1.2 Module contents

impracticalpythonprojects.

Example implementations of the projects in Impractical Python Projects.

MIT License

Jose A. Lerma III

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX