
impracticalpythonprojects

Release 0.10.1

Jose A. Lerma III

May 26, 2019

MODULE REFERENCE

1	src	3
1.1	src package	3
2	Indices and tables	9
	Python Module Index	11
	Index	13

Example implementations of the practice and challenge projects in [Impractical Python Projects](#). Alternative answers to practice projects and supporting files can be found at the [official GitHub page](#).

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).

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 `string.ascii_lowercase` as keys.

Raises `TypeError` – If `dictionary` is not a `dict`.

`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 `string.ascii_lowercase` as keys and a `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 pseudo-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 *str* or if **dictionary** isn't a *dict*.

`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 *dict*, and finally through *generate_name()* to make the actual name.

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

Returns String with pseudo-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 `add_keys_to_dict()`.

Type `str`

`src.ch01.challenge.SPLIT_NAME_LIST_ERROR`

String with `TypeError` for `split_names()`.

Type `str`

`src.ch01.challenge.SPLIT_NAME_EMPTY_ERROR`

String with `ValueError` for `split_names()`.

Type `str`

`src.ch01.challenge.ADD_NAME_TO_KEY_ERROR`

String with `TypeError` for `add_name_to_key()`.

Type `str`

`src.ch01.challenge.GENERATE_NAME_ERROR`

String with `KeyError` for `generate_name()`.

Type `str`

`src.ch01.challenge.BUILD_LIST_ERROR`

String with `IndexError` for `build_name_list()`.

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 in `VOWELS`, 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 `defaultdict` with each letter as keys and a `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.pprint()` to print dictionary with letter frequency analysis to terminal.

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

Returns `None`. 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 `encode()`.

Type `str`

`src.ch01.practice.FREQ_ANALYSIS_ERROR`

String with `TypeError` for Poor Bar Chart `freq_analysis()`.

Type `str`

`src.ch01.practice.PRINT_BAR_CHART_ERROR`

String with `TypeError` for Poor Bar Chart `print_bar_chart()`.

Type `str`

Module contents

Chapter 1.

src.ch02 package

Submodules

src.ch02.c1_recursive_palindrome module

Recursively determine if a word is a palindrome.

`src.ch02.c1_recursive_palindrome.main()`

Demonstrate the recursive palindrome tester.

`src.ch02.c1_recursive_palindrome.recursive_ispalindrome(word: str) → bool`

Recursively check if a word is a palindrome.

Parameters `word` (*str*) – String to check palindrome-ness.

Returns `True` if the word is a palindrome, `False` otherwise.

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

src.ch02.p1_cleanup_dictionary module

Remove single letter words from a word dictionary.

`src.ch02.p1_cleanup_dictionary.cleanup_dict(filepath: str) → list`
Wrap `read_from_file` and `cleanup_list`.

Passes given `filepath` through `read_from_file()` to get a list of words, then `cleanup_list()` to remove single letter words.

Parameters `filepath` (*str*) – String with path to word dictionary file.

Returns List with words as elements excluding single letter words.

`src.ch02.p1_cleanup_dictionary.cleanup_list(word_list: list) → list`
Cleanup word list.

Remove single letter words from a `list` of words.

Parameters `word_list` (*list*) – List with words as elements.

Returns List with words as elements excluding single letter words.

Raises `IndexError` – If `word_list` is empty.

`src.ch02.p1_cleanup_dictionary.main()`
Demonstrate cleanup dictionary.

Module contents

Chapter 2.

`src.ch02.DICTIONARY_FILE_PATH`
String with path to Ubuntu 18.04.2's American English dictionary file.

Type `str`

`src.ch02.CLEANUP_LIST_ERROR`
String with `IndexError` for Cleanup Dictionary `cleanup_list()`.

Type `str`

`src.ch02.RECURSIVE_ISPALINDROME_ERROR`
String with `TypeError` for Recursive Palindrome `recursive_ispalindrome()`.

Type `str`

1.1.2 Module contents

impracticalpythonprojects.

Example implementations of the projects in Impractical Python Projects.

MIT License

Jose A. Lerma III

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

S

- src, 8
- src.ch01, 7
- src.ch01.challenge, 5
- src.ch01.challenge.c1_foreign_bar_chart,
3
- src.ch01.challenge.c2_name_generator, 4
- src.ch01.practice, 7
- src.ch01.practice.p1_pig_latin, 6
- src.ch01.practice.p2_poor_bar_chart, 6
- src.ch02, 8
- src.ch02.c1_recursive_palindrome, 7
- src.ch02.p1_cleanup_dictionary, 8

A

ADD_KEYS_ERROR (in module *src.ch01.challenge*), 5
 add_keys_to_dict() (in module *src.ch01.challenge.c1_foreign_bar_chart*), 3
 add_name_to_key() (in module *src.ch01.challenge.c2_name_generator*), 4
 ADD_NAME_TO_KEY_ERROR (in module *src.ch01.challenge*), 5

B

BUILD_LIST_ERROR (in module *src.ch01.challenge*), 5
 build_name_list() (in module *src.ch01.challenge.c2_name_generator*), 4

C

cleanup_dict() (in module *src.ch02.p1_cleanup_dictionary*), 8
 cleanup_list() (in module *src.ch02.p1_cleanup_dictionary*), 8
 CLEANUP_LIST_ERROR (in module *src.ch02*), 8

D

DICTIONARY_FILE_PATH (in module *src.ch02*), 8

E

encode() (in module *src.ch01.practice.p1_pig_latin*), 6
 ENCODE_ERROR (in module *src.ch01.practice*), 7

F

foreign_freq_analysis() (in module *src.ch01.challenge.c1_foreign_bar_chart*), 3
 freq_analysis() (in module *src.ch01.practice.p2_poor_bar_chart*), 6
 FREQ_ANALYSIS_ERROR (in module *src.ch01.practice*), 7

G

generate_name() (in module *src.ch01.challenge.c2_name_generator*), 4
 GENERATE_NAME_ERROR (in module *src.ch01.challenge*), 5

M

main() (in module *src.ch01.challenge.c1_foreign_bar_chart*), 3
 main() (in module *src.ch01.challenge.c2_name_generator*), 4
 main() (in module *src.ch01.practice.p1_pig_latin*), 6
 main() (in module *src.ch01.practice.p2_poor_bar_chart*), 6
 main() (in module *src.ch02.c1_recursive_palindrome*), 7
 main() (in module *src.ch02.p1_cleanup_dictionary*), 8

N

name_generator() (in module *src.ch01.challenge.c2_name_generator*), 4

P

print_bar_chart() (in module *src.ch01.practice.p2_poor_bar_chart*), 6
 PRINT_BAR_CHART_ERROR (in module *src.ch01.practice*), 7

R

read_from_file() (in module *src.ch01.challenge.c2_name_generator*), 4
 recursive_ispalindrome() (in module *src.ch02.c1_recursive_palindrome*), 7
 RECURSIVE_ISPALINDROME_ERROR (in module *src.ch02*), 8

S

SPLIT_NAME_EMPTY_ERROR (in module *src.ch01.challenge*), 5

SPLIT_NAME_LIST_ERROR (in module
src.ch01.challenge), 5
split_names() (in module
src.ch01.challenge.c2_name_generator),
5
src (module), 8
src.ch01 (module), 7
src.ch01.challenge (module), 5
src.ch01.challenge.c1_foreign_bar_chart
(module), 3
src.ch01.challenge.c2_name_generator
(module), 4
src.ch01.practice (module), 7
src.ch01.practice.p1_pig_latin (module), 6
src.ch01.practice.p2_poor_bar_chart
(module), 6
src.ch02 (module), 8
src.ch02.c1_recursive_palindrome (mod-
ule), 7
src.ch02.p1_cleanup_dictionary (module), 8

V

VOWELS (in module src.ch01.practice), 7