djoseph-png / py-cat-and-dog-years

Code Pull requests Actions Projects Wiki Security Insights Settings

0 stars 1.5k forks 0 watching Branches Activity Tags

Public repository · Forked from mate-academy/py-cat-and-dog-years

1 Branch 0 Tags Go to file t Go to file Add file Code

This branch is up to date with `mate-academy/py-cat-and-dog-years:master` .

Contribute Sync fork

bvsn Merge pull request mate-academy#1434 from Serhii-Leonenko/extend_AI_c...

fa85a4c · 6 months ago

.github/workflows	Update test.yml	3 years ago
app	annotations to main.py added	3 years ago
tests	acheclist + more tests	3 years ago
.flake8	Update .flake8	3 years ago
.gitignore	Initial commit	3 years ago
.reviewrelatedfiles	added .reviewrelatedfiles	6 months ago
README.md	acheclist + more tests	3 years ago
checklist.md	Update checklist.md	2 years ago
requirements.txt	Update requirements.txt	3 years ago

README

Cat and Dog years

Read the guideline before starting.

Inside `app/test_main.py` , write tests for `get_human_age` function that takes two integers `cat_age` (my cat's age in cat years) and `dog_age` (my dog's age in dog years) and returns an array where:

- the first element is my cat's age in human years;
- the second element is my dog's age in human years.

As usually age is a whole number of years (discard the remainder).

Cat years are converted to human years following the next rules:

- first 15 cat years give 1 human year;
- the next 9 cat years give 1 more human year;
- every 4 next cat years give 1 extra human year.

Dog years:

- first 15 dog years give 1 human year;
- the next 9 dog years give 1 more human year;
- every 5 next dog years give 1 extra human year.

Please note: you have to use `pytest` for writing tests.

Examples:

```
get_human_age(0, 0) == [0, 0]
get_human_age(14, 14) == [0, 0]
get_human_age(15, 15) == [1, 1]
get_human_age(23, 23) == [1, 1]
get_human_age(24, 24) == [2, 2]
get_human_age(27, 27) == [2, 2]
get_human_age(28, 28) == [3, 2]
get_human_age(100, 100) == [21, 17]
```

Run `pytest app/` to check if function pass your tests.

Run `pytest --numprocesses=auto tests/` to check if your tests cover all boundary conditions and pass task tests.

Note: Check your code using this checklist before pushing your solution.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Python** 100.0%

Suggested workflows

Based on your tech stack

SLSA Generic generator Configure

Generate SLSA3 provenance for your existing release workflows

Python package Configure

Create and test a Python package on multiple Python versions.

Pylint Configure

Lint a Python application with pylint.

More workflows Dismiss suggestions