Kata

- Home
- Kata
- Kumite
- Forum
- Wiki

- Leaders

-
  - You have not starred any kata
    To add some, just click thenext to any kata title.

-
  - 2

    You have ranked up in Python!
    3 hours ago
  -
  - 2

    You have ranked up in C!
    19 hours ago
  -
  - 6 kyu

    Powerful you are. Your new overall rank is 6 kyu.
    2 weeks ago
  -
  - 7 kyu

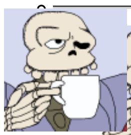
    Respect. Your new overall rank is 7 kyu.
    2 weeks ago
  -
  -

    100+ Honor: You now have the ability to weigh in on the ranking of beta kata!
    2 weeks ago
  -
  -

    75+ Honor: You now have the ability to estimate the ranking of your own beta kata.
    2 weeks ago
  -
  -

    50+ Honor: You now have the ability to mark comments as having spoiler code.
    2 weeks ago
  -
  -

    25+ Honor: You now have the ability to vote on beta kata.
    2 weeks ago
  -

- 
  6 kyu
  220
  - View Profile
  -
  - Account Settings
  - Training Setup
  - Upgrade to Red
  - Sign out

6 kyu

Banker's Plan

1091092982% of 369334 of 1,829g964

- Python
  Choose language...
    C
    Clojure
    CoffeeScript
    C++
    Crystal
    C#
    Elixir
    Forth (Beta)
    Fortran (Beta)
    F#
    Go
    Haskell
    Java
    JavaScript
    Julia (Beta)
    Kotlin (Beta)
    Nim (Beta)
    OCaml (Beta)
    PHP
    PowerShell (Beta)
    Prolog (Beta)
    Python
    R (Beta)
    Racket (Beta)
    Reason (Beta)
    Ruby
    Rust
    Scala (Beta)
    Shell
    Swift
    TypeScript
- 3.6.0
  Choose language version...
    2.7.6
    3.4.3
    3.6.0

-
  - VIM
  - EMACS
-
  -

  Instructions
  Output
Restore

- John has some amount of money of which he wants to deposit a part f0 to the bank at the beginning of year 1. He wants to withdraw each year for his living an amount c0.

  Here is his banker plan:

  - deposit f0 at beginning of year 1
  - his bank account has an interest rate of p percent per year, constant over the years
  - John can withdraw each year c0, taking it whenever he wants in the year; he must take account of an inflation of i percent per year in order to keep his quality of living. i is supposed to stay constant over the years.
  - all amounts $f_0..f_{n-1}$, $c_0..c_{n-1}$ are truncated by the bank to their integral part
  - Given f_0, p, c_0, i the banker guarantees that John will be able to go on that way until the nth year.

Example:

f0 = 100000, p = 1 percent, c0 = 2000, n = 15, i = 1 percent

beginning of year 2 -> f1 = 100000 + 0.01*100000 - 2000 = 99000; c1 = c0 + c0*0.01 = 2020 (with inflation of previous year)

beginning of year 3 -> f2 = 99000 + 0.01*99000 - 2020 = 97970; c2 = c1 + c1*0.01 = 2040.20
(with inflation of previous year, truncated to 2040)

beginning of year 4 -> f3 = 97970 + 0.01*97970 - 2040 = 96909.7 (truncated to 96909);
c3 = c2 + c2*0.01 = 2060.4 (with inflation of previous year, truncated to 2060)

and so on...

John wants to know if the bankers'plan is right or wrong. Given parameters f0, p, c0, n, i build a function fortune which returns true if John can make a living until the nth year and false if it is not possible.

Some cases:

```
fortune(100000, 1, 2000, 15, 1) -> True
fortune(100000, 1, 10000, 10, 1) -> True
fortune(100000, 1, 9185, 12, 1) -> False
```

For the last case you can find below the amounts of his account at the beginning of each year:
100000, 91815, 83457, 74923, 66211, 57318, 48241, 38977, 29523, 19877, 10035, -5

f_{11} = -5 so he has no way to withdraw something for his living in year 12.

> **Note:** Don't forget to convert the percent parameters as percentages in the body of your function: if a parameter percent is 2 you have to convert it to 0.02.

Algorithms
Mathematics
Numbers

powered by

Solution:

```
xxxxxxxxxx
def fortune(f0, p, c0, n, i):
    # your code
```

Sample Tests:

Your output will be shown here

```
Your results will be shown
here.
```

•
•
•
•
•
•
•

xxxxxxxxxx

SkipUnlock SolutionsDiscuss (26)Reset
TestAttemptSubmit

```
@test.describe('Tests')
def fixed_tests():

    @test.it('Basic Tests')
    def tests():
        Test.assert_equals(fortune(100000, 1, 2000, 15, 1), True)
        Test.assert_equals(fortune(100000, 1, 9185, 12, 1), False)
        Test.assert_equals(fortune(100000000, 1, 100000, 50, 1), True)
        Test.assert_equals(fortune(100000000, 1.5, 10000000, 50, 1), False)
        Test.assert_equals(fortune(100000000, 5, 1000000, 50, 1), True)

```