Descrição | Visualizar envios

EP06

Data de entrega: segunda, 14 jun 2021, 23:59
Arquivos requeridos: funcoes.py (Baixar)
Tipo de trabalho: Trabalho individual
Redução por avaliação automática: 0.5 **Avaliações livres**: 16

EP06 - funções

Fonte: 'Calvin e Haroldo', por Bill Watterson

O **limite de submissões livres** deste EP é 16.

O **prazo de entrega** deste EP é 23h 59m do dia 09/06/2021. O sistema reabre a partir do dia 10/06 às 12h para envio de EPs com atraso, por mais 5 dias, recebendo desconto de 2 pontos por dia.

Objetivos

Praticar a construção de funções em Python.

Introdução

Neste exercício você escreverá quatro função: `primo()`, `goldbach()`, `exponencial()` e `logistica()`. Adiante é descrito o comportamento de cada função e alguns exemplos. O cabeçalho das funções está o arquivo `funcoes.py` que deve ser baixado a partir desta página. Sugerimos que depois de escrever e testar cada função você submeta seu EP para avaliação.

`primo()`

A função `primo()` recebe um número inteiro `n` e retorna `True` se `n` é um número primo e `False` em caso contrário.

> Um número inteiro maior que 1 é **primo** se seus únicos divisores positivos são 1 e ele mesmo.

Exemplos no Python Shell

A seguir estão alguns exemplos de execução da função no Python Shell.

```
In [19]: primo(1)
Out[19]: False

In [20]: primo(2)
Out[20]: True

In [21]: primo(27644437)
Out[21]: True

In [22]: primo(327183)
Out[22]: False

In [23]: primo(2321)
Out[23]: False

In [24]: primo(-5)
Out[24]: False
```

Depois de escrever e testar a função `primo()`, submeta seu EP para avaliação.

---

goldbach()

Esta função é relacionada com a Conjectura de Goldbach.

A função `goldbach()` recebe um inteiro positivo `n` e se `n` é soma de dois primos `p` e `q` a função retorna `True`, `p` e `q`. Em caso contrário a função retorna `False`, `1` e `1`.

Para, por exemplo, sua função retornar `False`, `1` e `1`, basta escrever

```
return False, 1, 1
```

Aqui você pode simular uma função exemplo que retorna 2 valores.

Exemplos no Python Shell

A seguir estão alguns exemplos de execução da função no Python Shell.

```
In [50]: b1, p1, q1 = goldbach(1)

In [51]: b1
Out[51]: False

In [52]: q1
Out[52]: 1

In [53]: p1
Out[53]: 1

In [54]: b8, p8, q8 = goldbach(8)

In [55]: print(b8, p8, q8)
True 3 5

In [56]: b12, p12, q12 = goldbach(12)

In [57]: print(b12, p12, q12)
True 5 7

In [58]: b15, p15, q15 = goldbach(15)

In [59]: print(b15, p15, q15)
True 2 13

In [60]: b1248, p1248, q1248 = goldbach(1248)

In [61]: print(b1248, p1248, q1248)
True 11 1237
```

Depois de escrever e testar a função `goldbach()`, submeta seu EP para avaliação.

---

exponencial()

Não é necessário, mas caso você tenha curiosidade, o vídeo em que se baseou esta função pode ser visto no final deste enunciado.

A função `exponencial()` recebe como parâmetros:

- o número `n0` de indivíduos infectados em um dia inicial que chamaremos de dia 0;
- o número diário médio `e` de indivíduos com quem alguém infectado é exposto;
- a probabilidade `p` de uma exposição resultar em uma infecção;
- um inteiro `d`, `d ≥ 0`.

A função retorna o número total de indivíduos infectados até o dia `d` **determinado por** `(n0, e, p)`. Esse valor é calculado da seguinte maneira:

- no dia `0` o número de infectados é n_0;
- no dia `1` o número de infectados é $n_1 = n_0 + e \times p \times n_0 = (1 + e \times p) \times n_0$;
- no dia `2` o número de infectados é $n_2 = n_1 + e \times p \times n_1 = (1 + e \times p) \times n_1 = (1 + e \times p)^2 \times n_0$;
- no dia `3` o número de infectados é $n3 = n_2 + e \times p \times n_2 = (1 + e \times p) \times n_2 = (1 + e \times p)^3 \times n_0$;
- ...
- no dia `d` o número de infectados é $n_d = (1 + e \times p)^d \times n_0$.

Para estarmos tod@s calculando os mesmos valores faça o seguinte:

> todos valores intermediários computados pela função **devem ser** `float`, entretanto o valor retornado pela função **deverá ser** `int`.

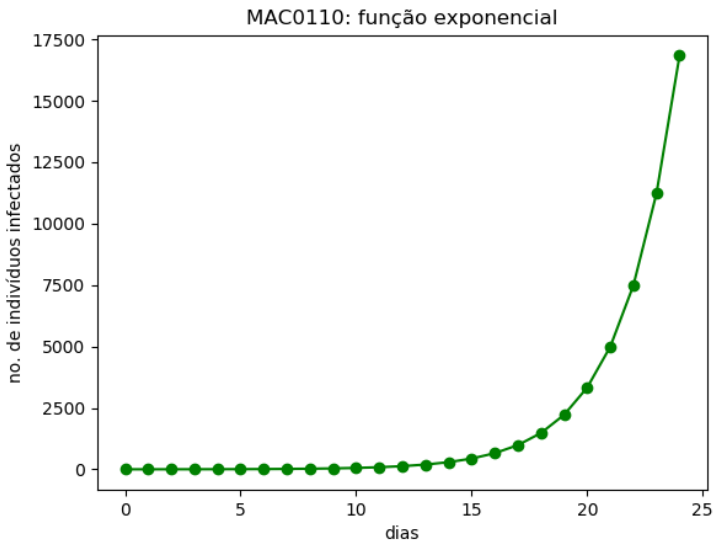
Para essa conversão de `float` para `int` utilize a função `int()` do Python.

```
In [62]: int(5.0)
Out[62]: 5

In [63]: int(5)
Out[63]: 5

In [64]: int(5.7)
Out[64]: 5

In [65]: int(5.99)
Out[65]: 5
```

O distanciamento físico entre indivíduos diminui o valor de `e`. Práticas de higiene como lavar as mãos diminuem a probabilidade `p`.

Gráfico

Veja um gráfico da função para `n0 = 1`, `e = 5` e `p = 0.1`.



Exemplos de cálculos

Suponha que `n0 = 1`, `e = 5` e `p = 0.1`. Então para o dia

- `d = 0` o número de infectados é `n0 = 1`;
- `d = 1` o número de infectados é `n1 = (1 + 5*0.1)*1 = 1.5`, logo `n1 = 1`.
- `d = 2` o número de infectados é `n2 = (1 + 5*0.1)*(1 + 5*0.1)*1 = 2.25`, logo `n2 = 2`.
- `d = 3` o número de infectados é `n3 = (1 + 5*0.1)*(1 + 5*0.1)*(1 +5*0.1)*1 = 3.375`, logo `n3 = 3`.

Exemplos no Python Shell

A seguir estão alguns exemplos de execução da função no Python Shell.

```
In [66]: exponencial(1, 5, 0.1, 0)
Out[66]: 1

In [67]: exponencial(1, 5, 0.1, 1)
Out[67]: 1

In [68]: exponencial(1, 5, 0.1, 2)
Out[68]: 2

In [69]: exponencial(1, 5, 0.1, 5)
Out[69]: 7

In [70]: exponencial(1, 5, 0.1, 20)
Out[70]: 3325

In [71]: exponencial(1, 5, 0.1, 30)
Out[71]: 191751

In [72]: exponencial(1, 5, 0.1, 40)
Out[72]: 11057332
```

Depois de escrever e testar a função `exponecial()`, submeta seu EP para avaliação.

`logistica()`

Não é necessário, mas caso você tenha curiosidade, o vídeo em que se baseou esta função pode ser visto no final deste enunciado.

A função logistica leva em consideração o fato de que na prática funções exponenciais não são encontradas. No exemplo do surto por coronavírus temos que o número de indivíduos infectados não pode crescer indefinidamente. Um limite óbvio é o número `n` de indivíduos na população. Esse valor `n` é utilizado pela função `logistica()` que inicialmente tem um comportamento muito próximo da função `exponencial()`

A função `logistica()` recebe como parâmetros:

- o número `n0` de indivíduos infectados em um dia inicial que chamaremos de dia 0;
- o número diário médio `e` de indivíduos com quem alguém infectado é exposto;
- a probabilidade `p` de uma exposição resultar em uma infecção;
- o número `n` de indivíduos na população; e
- um inteiro `d`, $d \geq 0$.

A função retorna o número total de indivíduos infectados até o dia `d` **determinado por** `(n0, e, p, n)`. Esse valor é calculado da seguinte maneira:

- no dia `0` o número de infectados é n_0;
- no dia `1` o número de infectados é $n_1 = n_0 + e \times p \times (1 - \frac{n_0}{n}) \times n_0 = (1 + e \times p \times (1 - \frac{n_0}{n})) \times n_0$;
- no dia `2` o número de infectados é $n_2 = n_1 + e \times p \times (1 - \frac{n_1}{n}) \times n_1 = (1 + e \times p \times (1 - \frac{n_1}{n})) \times n_1$;
- no dia `3` o número de infectados é $n_3 = n_2 + e \times p \times (1 - \frac{n_2}{n}) \times n_2 = (1 + e \times p \times (1 - \frac{n_2}{n})) \times n_2$;
- ...
- no dia `d` o número de infectados é $n_d = (1 + e \times p \times (1 - \frac{n_{d-1}}{n})) \times n_{d-1}$.

Para estarmos tod@s calculando os mesmos valores faça o seguinte:

> todos valores intermediários computados pela função **devem ser** `float`, entretanto o valor retornado pela função **deverá ser** `int`.

Para essa conversão de `float` para `int` utilize a função `int()` do Python.

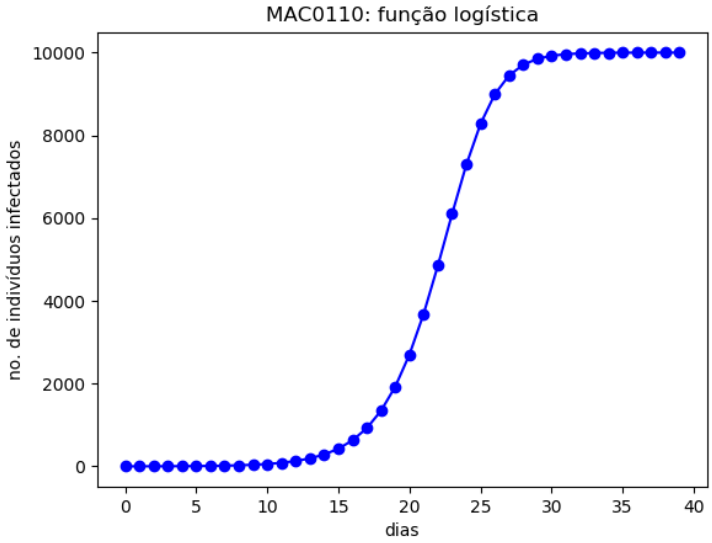
Exemplos de cálculos

Suponha que `n0 = 1`, `e = 5`, `p = 0.1` e n=10000. Então para o dia

- `d = 0` o número de infectados é `n0 = 1`;
- `d = 1` o número de infectados é `n1 = (1 + 5*0.1*(1 - 1/10000))*1 = 1.499`, logo `n1 = 1`.
- `d = 2` o número de infectados é `n2 = (1 + 5*0.1*(1 - 1.499/10000))*1.499 = 2.248`, logo `n2 = 2`.
- `d = 3` o número de infectados é `n3 = (1 + 5*0.1*(1 - 2.248/10000))*2.248 = 3.371`, logo `n3 = 3`.
- `d = 4` o número de infectados é `n4 = (1 + 5*0.1*(1 - 3.371/10000))*3.371 = 5.055`, logo `n4 = 5`.

Gráfico

Veja um gráfico da função para `n_0 = 1`, `e = 5`, `p = 0.1` e `n=10000`.



Exemplos no Python Shell

A seguir estão alguns exemplos de execução da função no Python Shell.

```
In [77]: logistica(1, 5, 0.1, 10000, 0)
Out[77]: 1

In [78]: logistica(1, 5, 0.1, 10000, 1)
Out[78]: 1

In [79]: logistica(1, 5, 0.1, 10000, 2)
Out[79]: 2

In [80]: logistica(1, 5, 0.1, 10000, 3)
Out[80]: 3

In [81]: logistica(1, 5, 0.1, 10000, 4)
Out[81]: 5

In [82]: logistica(1, 5, 0.1, 10000, 10)
Out[82]: 57

In [83]: logistica(1, 5, 0.1, 10000, 20)
Out[83]: 2701

In [84]: logistica(1, 5, 0.1, 10000, 30)
Out[84]: 9923

In [85]: logistica(1, 5, 0.1, 10000, 40)
Out[85]: 9999
```

Depois de escrever e testar a função `logistica()`, submeta seu EP para avaliação.

Roteiro

- **Baixe** o arquivo `funcoes.py` para uma pasta no computador que estiver usando. Este é o único arquivo que deverá ser depositado nesta página.
- **Leia** o cabeçalho com atenção e **preencha** o seu nome e número USP.
- **Implemente** e **teste** cada função **separadamente**.
- Depois de testar uma função, **deposite o seu EP** na página da disciplina para que a **função seja avaliada**.

Este EP pode ser enviado 16 vezes sem prejuízo de nota. A partir da 17a submissão, haverá um desconto de 0.5, meio ponto, por submissão. Verifique as mensagem do programa `avaliador` e, caso necessário, faça eventuais correções e ressubmeta o seu EP. Evite submissões desnecessárias, para não ultrapassar o limite de submissões. Procure depurar o seu programa no seu computador, usando o Spyder.

Submissões feitas após às 23h 59m do dia 09/06/2021 receberão desconto de 2 pontos por dia de atraso. O EP não poderá ser enviado após 14/06/2021.

Honestidade Acadêmica

Esse é um exercício individual, não em grupo. Isso não significa que você não pode receber ajuda de outras pessoas, inclusive de seus colegas. De uma forma geral, gostaríamos de incentivar as discussões de ideias, conceitos e alternativas de solução. Nossa maior recomendação é evitar olhar o código fonte de uma solução antes de escrever o seu programa. Em caso de dúvida, consulte nossa política de colaboração.

De forma sucinta, evite as seguintes ações que caracterizam desonestidade acadêmica na realização dos trabalhos individuais desta disciplina:

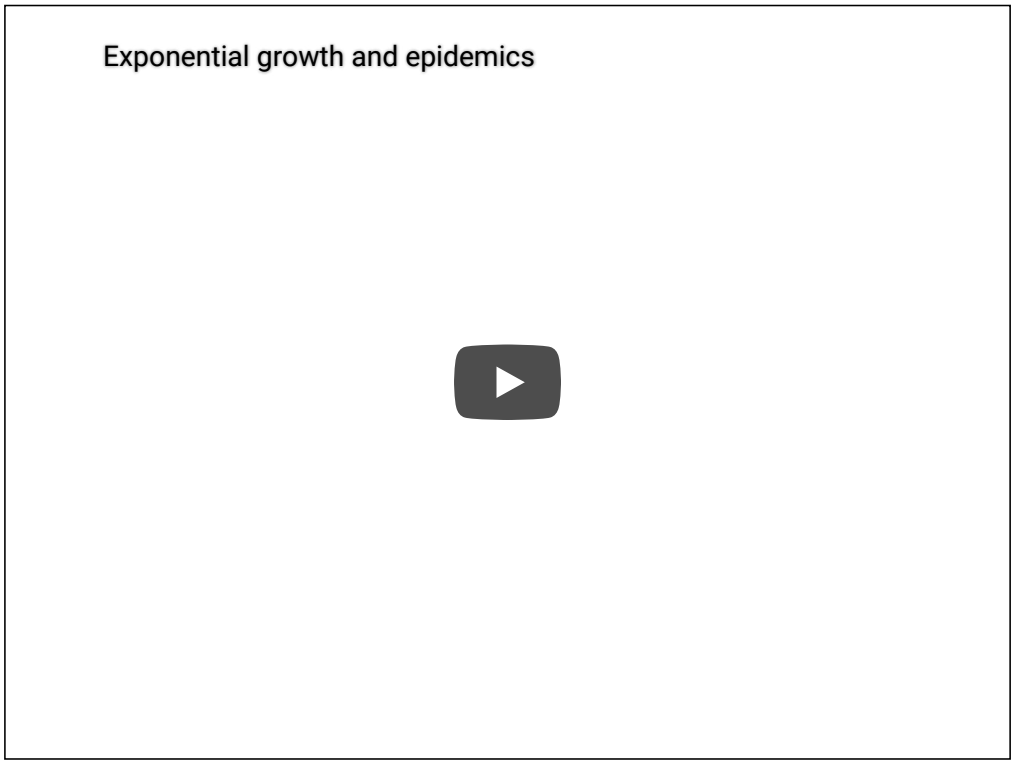
- buscar e obter uma solução, parcial ou completa, correta ou não, de um EP na internet ou qualquer outro meio físico ou virtual, durante o período de submissão do referido EP;
- solicitar ou obter uma cópia, parcial ou completa, correta ou não, da solução de um EP durante o seu período de submissão;
- permitir que um colega acesse uma cópia, parcial ou completa, correta ou não, do seu EP, durante o período de submissão;
- ainda mais grave é o plágio, que se configura pela utilização de qualquer material não visto em aula ou não descrito no enunciado, que não seja de sua autoria, em parte ou ao todo, e entregar, com ou sem edição, como se fosse seu trabalho, para ser avaliado.

Curiosidades

O vídeo em que baseou as funções `exponencial()` e `logistica()` teve como objetivo apenas ilustrar o significado do *crescimento exponencial* e *logístico* de funções. O vídeo simplesmente utilizou como motivação função que dá o número de indivíduos infectados pelo vírus *covid-19*. Não é algo feito por, digamos, epidemiologistas. A criação de modelos preditivos para surtos como o do coronavírus exigem uma colaboração muito grande entre cientistas de *diversas áreas* tais como biologia, ciência social, engenharia, epidemiologia, estatística, matemática, medicina, saúde pública, química,... Visite a página do *Imperial College COVID-19 Response Team* (MRC Centre for Global Infectious Diseases Analyses). Os relatórios desse grupo têm sido muito citados pela imprensa, governos, profissionais de saúde ... ao redor do mundo.

A propósito, um artigo muito bacana sobre o popular *achatamento da curva* fui publicado pelo Washington Post. A versão original em inglês do artigo pode ser vista aqui. Uma tradução para português pode ser vista aqui.

O vídeo em que baseou as funções `exponencial()` e `logistica()` foi



.

Arquivos requeridos

funcoes.py

```python
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------
# LEIA E PREENCHA O CABEÇALHO
# NÃO ALTERE OS NOMES DAS FUNÇÕES, MÉTODOS E ATRIBUTOS
# NÃO APAGUE OS DOCSTRINGS
# ---------------------------------------------------------------------

"""
    Nome:
    NUSP:

    Ao preencher esse cabeçalho com o meu nome e o meu número USP,
    declaro que todas as partes originais desse exercício programa (EP)
    foram desenvolvidas e implementadas por mim e que portanto não
    constituem desonestidade acadêmica ou plágio.
    Declaro também que sou responsável por todas as cópias desse
    programa e que não distribui ou facilitei a sua distribuição.
    Estou ciente que os casos de plágio e desonestidade acadêmica
    serão tratados segundo os critérios divulgados na página da
    disciplina.
    Entendo que EPs sem assinatura devem receber nota zero e, ainda
    assim, poderão ser punidos por desonestidade acadêmica.

    Abaixo descreva qualquer ajuda que você recebeu para fazer este
    EP.  Inclua qualquer ajuda recebida por pessoas (inclusive
    monitores e colegas). Com exceção de material de MAC0110, caso
    você tenha utilizado alguma informação, trecho de código,...
    indique esse fato abaixo para que o seu programa não seja
    considerado plágio ou irregular.

    Exemplo:

        A monitora me explicou que eu devia utilizar a função int() quando
        fazemos leitura de números inteiros.

        A minha função quicksort() foi baseada na descrição encontrada na
        página https://www.ime.usp.br/~pf/algoritmos/aulas/quick.html.

    Descrição de ajuda ou indicação de fonte:

"""

#---------------------------------------------------
def primo(n):
    '''(int) -> bool

       RECEBE um número inteiro n.
       RETORNA True se n é primo e False em caso contrário.
    '''
    # remova o print() e escreva suas função a seguir
    print("Vixe! Ainda não fiz a função primo()")

#---------------------------------------------------
def goldbach(n):
    '''(int) -> bool, int, int

       RECEBE um número inteiro n.
       RETORNA True, p, q se n é soma de dois números primos p e q.
       Se n não é soma de dois números primos a função retorna False, 1, 1.
    '''
    # remova o print() e escreva suas função a seguir
    print("Vixe! Ainda não fiz a função goldebach()")

#---------------------------------------------------
def exponencial(n0, e, p, d):
    '''(int, int, float, int) -> int

       RECEBE

         * o número n0 de indivíduos infectados em um dia 0;
         * o número diário médio e de indivíduos com quem alguém
           infectado é exposto;
         * a probabilidade p de uma exposição resultar em uma infecção;
         * um inteiro d,  d >=  0.

       RETORNA o número total de indivíduos infectados até o dia d
       determinado por (n0, e, p).
    '''
    print("Vixe! Ainda não fiz a função exponencial()")

#---------------------------------------------------
def logistica(n0, e, p, n, d):
    '''(int, int, float, int, int) -> int

       RECEBE

         * o número n0 de indivíduos infectados em um dia 0;
         * o número diário médio e de indivíduos com quem alguém
           infectado é exposto;
         * a probabilidade p de uma exposição resultar em uma infecção;
         * o número n de indivíduos na população; e
         * um inteiro d, d >= 0.
```

```
94          RETORNA o número total de indivíduos infectados até o dia d
95          determinado por (n0, e, p, n).
96
97      '''
98      print("Vixe! Ainda não fiz a função logistica()")
99
```

VPL

◄ EP07

Seguir para...

EP05 ►