Descrição | Visualizar envios

EP07

Disponível a partir de: quinta, 10 jun 2021, 00:01
Data de entrega: quarta, 16 jun 2021, 23:59
Arquivos requeridos: area.py (Baixar)
Tipo de trabalho: Trabalho individual
Redução por avaliação automática: 0.5 **Avaliações livres**: 10

EP07 - Aproximações de áreas

Fonte: https://novaescola.org.br/conteudo/3621/calvin-e-seus-amigos

O **limite de submissões livres** deste EP é 10.

O **prazo de entrega** deste EP é 23h 59m do dia 16/06/2021. O sistema reabre a partir do dia 17/06 às 12h para envio de EPs com atraso, por mais 5 dias, recebendo desconto de 2 pontos por dia.

Sugerimos que depois de escrever e testar cada função você submeta seu EP para avaliação.

Objetivos

- Treinar mais ainda a definição e uso de funções: funções como parâmetros, retorno de vários valores e argumento opcional.
- Manipular valores `float` que são representações aproximadas em ponto flutuante de número reais.
- Obter valores aproximados de uma função.
- Usar a biblioteca `math`.
- Praticar o uso de módulos em Python.
- Fazer experimentos envolvendo teoria do erro através dos conceito de erro absoluto e erro relativo.

Área sob uma função

Suponha que f é uma função com valores ≥ 0 e contínua (= sem 'buracos') no intervalo $[a, b]$. Portanto, $f(x) < M$ para todo $x \in [a, b]$ para alguma número M. A região hachurada na figura abaixo corresponde à área da região A de f nesse intervalo.

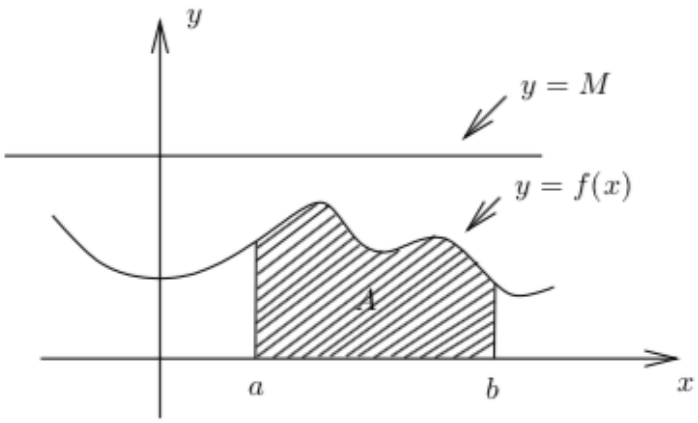


Figura 1: Região sob *f*.

Nesse exercício vamos utilizar o **método dos retângulos** para obter uma aproximação da área da região *A*. A ideia é simples e consiste em

- *particionar* o intervalo em *k* subintervalos de mesmo comprimento,
- *aproximar* a área de *A* em cada um desses subintervalos através da área de um retângulo,
- *calcular* a área de cada retângulo, e
- *adicionar* a área de todos esses retângulos, obtendo assim uma apróximação da área da região *A*.

O que dá nome ao método é a utilização *retângulos* para dividir a região *A*. Há outros métodos que dividem a região em outras formas geométricas, como trapézios. No *limte* todos esses métodos produzem um valor que se aproxima da área a medida que *k* cresce. A figura abaixo ilustra essa ideia para $k = 3$.

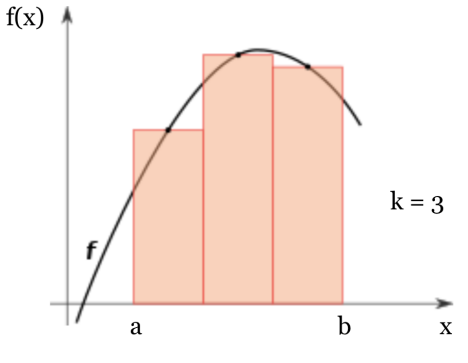
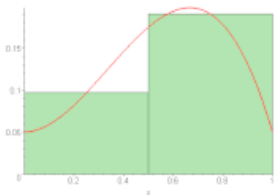


Figura 2: Método dos retângulos.

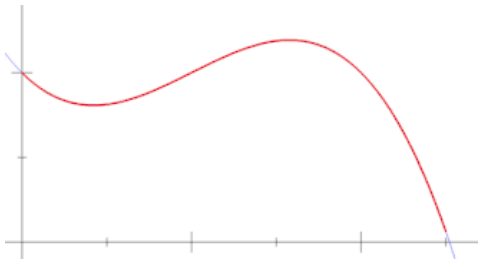
Tome $\Delta_x = (b - a)/k$ como sendo a **base** de cada retângulo. Observe que a área do primeiro retângulo, mais a esquerda, pode ser calculada como $f(x_{meio}) \cdot \Delta_x$, onde x_{meio} é o ponto médio do subintervalo $[a, a + \Delta_x]$. Portanto, a área da função no intervalo $[a, b]$ pode ser aproximada pelo somatória de todos os retângulos de base Δ_x e **altura** dada pelo valor de *f* no ponto x_{meio}, valor *x* no meio do retângulo.

É evidente que a precisão do resultado depende do número *k* de retângulos. Quanto maior o número *k* de retângulos, mais próximo estará o valor calculado do valor real da área da região sob *f* no intervalo $[a, b]$.



Fonte: Riemann sum (Wikipedia)

A animação abaixo ilustra uma sequência de somas para vários valores de *k*. O número no canto superior esquerdo indica a área total dos retângulos. Esse valor *converge* para o valor da área sob a função no intervalo a medida que *k* aumenta.



Fonte: Riemann integral (Wikipedia)

Erro Relativo

Dado um número *x* e uma aproximação *y* para *x*, o `erro`, também chamado de **erro absoluto**, da aproximação *y* em relação a *x* é definido como sendo |*y* − *x*|. Quando a grandeza de *x* não é próxima de 1, o erro absoluto pode não ser a maneira mais adequada de medir a qualidade da aproximação *y*.
Por exemplo, os erros absolutos de 1.01 em relação a 1.00 e de 0.02 em relação a 0.01 são idênticos, mas a nossa intuição nos diz que a primeira aproximação é muito melhor que a segunda. Faz sentido?.

Face à limitada avaliação de uma aproximação conferida pelo erro absoluto, tenta-se definir o **erro relativo** de *y* em relação a *x* como sendo |(*y* − *x*)/*x*|, em que |*z*| indica o valor absoluto de *z*; o valor de |1.7| e | − 1.7| é 1.7.

Assim, nos dois exemplos anteriores, os erros relativos são respectivamente de 0.01 (ou 1%) e 1 (ou 100%). Contudo, esta definição ainda é incompleta quando $x = 0$. Nestes casos, para se evitar a divisão por 0, adotam-se valores arbitrários para o erro relativo.

No caso de também ocorrer que $y = 0$, a aproximação certamente é perfeita e adota-se que o erro é 0. No caso de $y \neq 0$, a aproximação é certamente insatisfatória e adota-se o valor arbitrário 1 para o erro relativo. Na prática, os erros relativos procurados são sempre menores que 1.

Resumindo, tem-se que

- `erro_rel(y,x)` = 0, **se** `x = 0` e `y = 0`,
- `erro_rel(y,x)` = 1, **se** `x = 0` e `y ≠ 0`, e
- `erro_rel(y,x)` = `|(y - x)/x|`, **se** `x ≠ 0`.

Aproximação da área

Vimos que, aumentando o número `k` de retângulos, obtemos uma aproximação melhor da área. Como sabermos qual o valor de `k` que devemos usar para obtermos um aproximação com uma precisão *desejada* ou *aceitável*?

A aceitabilidade de uma aproximação depende da aplicação. No momento vamos tomar como medida de precisão um valor *pequeno* que apelidaremos de `EPSILON`. Quanto menor `EPSILON`, melhor é a qualidade da aproximação.

Uma forma de garantir que o valor da aproximação da área esteja dentro da precisão desejada é aumentar o valor de `k` até que o *erro relativo* entre duas aproximações sucessivas seja menor que o nosso desejo de precisão `EPSILON`.

O que você deve fazer

Nesse exercício você deve implementar as seguintes duas funções:

```
#-----------------------------------------------------------------
def area_por_retangulos(f, a, b, k):
    '''(function, float, float, int) -> float

    RECEBE uma função f, dois números a e b e um inteiro k.
    RETORNA uma aproximação da área sob a função f no intervalo [a,b]
        usando k retângulos.
    PRÉ-CONDIÇÃO: a função supõe que a função f é continua no intervalo [a,b] e que
          f(x) >= 0 para todo x, a <= x <= b.
    '''
    # corpo da função
    return aproximacao # nome ilustrativo

#-----------------------------------------------------------------
def area_aproximada(f, a, b, eps=EPSILON):
    '''(function, float, float, float) -> int, float

    RECEBE uma função f, dois números a, b, eps.
    RETORNA um inteiro k e uma aproximação da área sob a função f no intervalo [a,b]
        usando k retângulo.

    O valor de k deve ser a __menor potência__ de 2 tal que o erro relativo
    da aproximação retornada seja menor que eps.

    Assim, os possíveis valores de k são 1, 2, 4, 8, 16, 32, 64, ...

    PRÉ-CONDIÇÃO: a função supõe que a função f é continua no intervalo [a,b] e que
          f(x) >= 0 para todo x, a <= x <= b.
    '''
    # coprpo da função
    return k, aproximacao # nomes ilustrativos
```

A função `area_aproximada()` deve chamar a `area_por_retangulos(f, a, b, k)` com valores de `k` cada vez maiores, calculando o erro relativo entre duas chamadas consecutivas, até que o erro relativo seja estritamente menor que a precisão `eps` desejada.

Para melhorar o desempenho da função `area_aproximada()`, em vez de, ao final de cada iteração, incrementar de 1 o valor de `k`, sua função deve **dobrar** o valor de `k`. Desta forma, `k` será uma potencia de 2 e assumirá os valores na sequência 1, 2, 4, 8, 16, 32, 64, 128, ...

Sua solução deverá usar a função `erro_rel(y, x)`, que calcula e retorna o erro relativo entre `y` e `x`, que já está implementada no arquivo `area.py`.

Aproximação da área sob uma função qualquer

As duas funções que você deve implementar recebem uma função `f()` como argumento, da mesma forma que recebe os valores como `a`, `b` e `k` ou, opcionalmente, `eps`. Passagem de funções como argumento é um recurso de Python que vamos explorar para aproximar a área de qualquer função `f(x)` que recebe e retorna um valor real não negativo.

Veja está simulação aqui para ajudar entender o mecanismo de passagem de uma função como parâmetro.

O arquivo `area.py` contém um exemplo de função `main()` que mostra como utilizar as funções `area_por_retangulos()` e `area_aproximada()`. A função `main()` também mostra como fazer referência a funções definidas no próprio programa com `def ...`. Mostra ainda como fazer referência a uma função de algum outro módulo como o `math`; neste caso necessitamos adicionar uma linha com `import math`.

O arquivo `area.py` contém ainda três funções auxiliares: `identidade(x)`, `circunferencia(x)` e `exp(x)`.
Estude essas funções para aprender a escrever suas próprias funções matemáticas de uma variável e utilize-as para testar as suas funções para cálculo da aproximação da área.

Para aproximar a área de outras funções, você deve modificar a seguinte atribuição na função `main()`:

```
    f_x = identidade # f_x passa a ser um apelido de identidade
```

Nessa atribuição, a variável `f_x` passa a **fazer referência** ou **ser um apelido** à função `identidade()`, que é uma das funções definidas no programa. Observe ainda na `main()` como a variável `f_x` é passada como argumento nas chamadas das funções `area_por_retangulos()` e `area_aproximada()`.

A variável `f_x` é conveniente para que possamos testar o cálculo da aproximação da área de diferentes funçõesalterando apenas esta atribuição, pois poderíamos passar o valor `identidade` diretamente nas chamadas como por exemplo `area_por_retangulos(identidade, 0, 1, 10)` em vez de usar variáveis como em `area_por_retangulos(f_x, a, b, k)`.

Módulo `math`

Além de aprender a passar uma função como parâmetro de outra função, vamos trabalhar nesse exercício com números em ponto flutuante e, portanto, com *resultados aproximados*.

Esse também é um exercício de exploração do módulo `math` do Python, que contém várias rotinas matemáticas como:

- `math.sin(x)`: calcula e retorma o valor de `sen(x)`;

- `math.cos(x)`: calcula e retorma o valor de `cos(x)`;
- `math.exp(x)`, calcula e retorna o valor de e^x;

Para mais informações, no `iPython` escreva

```
In [1]: import math

In [2]: help(math)
[... montes de funções e informações ...]
```

Para calcular a área sob outras funções, modifique a variável `f_x` na função `main()` para fazer referência a estas outras funções, como algumas do módulo `math`, escrevendo por exemplo:

```
f_x = math.cos
a = 0 # experimente também outros intervalos
b = math.pi
```

Exemplos

A seguir estão as saídas produzidas pela função `main()` que está em `area.py`

Saída para a `identidade()`

Após implementar essas funções, a saída esperada da função `main()` com o comando `f_x = identidade` é:

```
Início dos testes.
A função f_x usada nos testes é identidade()
Valor de f_x(0.0)= 0.0
Valor de f_x(0.5)= 0.5
Valor de f_x(1.0)= 1.0

Área por retângulos:
teste 1: para 1 retângulos no intervalo [0, 1]:
    área aproximada = 0.5
teste 2: para 10 retângulos no intervalo [0, 1]:
    área aproximada = 0.5
teste 3: para 100 retângulos no intervalo [0, 1]:
    área aproximada = 0.5

Área aproximada:
teste 1: para eps = 1e-06 e intervalo [0, 1]:
     com 2 retângulo a área é aproximadamente = 0.5
teste 2: para eps = 1e-05 e intervalo [0, 1]:
     com 2 retângulos a área é aproximadamente = 0.5
teste 3: para eps = 0.0001 e intervalo [0, 1]:
     com 2 retângulos a área é aproximadamente = 0.5
teste 4: para eps = 0.001 e intervalo [0, 1]:
     com 2 retângulos a área é aproximadamente = 0.5
Fim dos testes.
```

Saída para a `circunferencia()`

A saída da função `main()` com o comando `f_x = circunferencia` é:

```
Início dos testes.
A função f_x usada nos testes é circunferencia()
Valor de f_x(0.0)= 1.0
Valor de f_x(0.5)= 0.8660254037844386
Valor de f_x(1.0)= 0.0

Área por retângulos:
teste 1: para 1 retângulos no intervalo [0, 1]:
    área aproximada = 0.866025
teste 2: para 10 retângulos no intervalo [0, 1]:
    área aproximada = 0.788103
teste 3: para 100 retângulos no intervalo [0, 1]:
    área aproximada = 0.785484

Área aproximada:
teste 1: para eps = 1e-06 e intervalo [0, 1]:
     com 4096 retângulo a área é aproximadamente = 0.785398
teste 2: para eps = 1e-05 e intervalo [0, 1]:
     com 1024 retângulos a área é aproximadamente = 0.785401
teste 3: para eps = 0.0001 e intervalo [0, 1]:
     com 256 retângulos a área é aproximadamente = 0.785419
teste 4: para eps = 0.001 e intervalo [0, 1]:
     com 64 retângulos a área é aproximadamente = 0.785566
Fim dos testes.
```

Saída para a `exp()`

A saída da função `main()` para `f_x = exp` é:

```
Início dos testes.
A função f_x usada nos testes é exp()
Valor de f_x(0.0)= 1.0
Valor de f_x(0.5)= 1.6487212707001282
Valor de f_x(1.0)= 2.718281828459045

Área por retângulos:
teste 1: para 1 retângulos no intervalo [0, 1]:
    área aproximada = 1.64872
teste 2: para 10 retângulos no intervalo [0, 1]:
    área aproximada = 1.71757
teste 3: para 100 retângulos no intervalo [0, 1]:
    área aproximada = 1.71827

Área aproximada:
teste 1: para eps = 1e-06 e intervalo [0, 1]:
     com 512 retângulo a área é aproximadamente = 1.71828
teste 2: para eps = 1e-05 e intervalo [0, 1]:
     com 128 retângulos a área é aproximadamente = 1.71828
teste 3: para eps = 0.0001 e intervalo [0, 1]:
     com 64 retângulos a área é aproximadamente = 1.71826
teste 4: para eps = 0.001 e intervalo [0, 1]:
     com 16 retângulos a área é aproximadamente = 1.718
Fim dos testes.
```

Saída para math.cos

A saída da função `main()` para `f_x = math.cos` é:

```
Início dos testes.
A função f_x usada nos testes é cos()
Valor de f_x(0.0)= 1.0
Valor de f_x(0.5)= 0.8775825618903728
Valor de f_x(1.0)= 0.5403023058681398

Área por retângulos:
teste 1: para 1 retângulos no intervalo [0, 1]:
    área aproximada = 0.877583
teste 2: para 10 retângulos no intervalo [0, 1]:
    área aproximada = 0.841822
teste 3: para 100 retângulos no intervalo [0, 1]:
    área aproximada = 0.841474

Área aproximada:
teste 1: para eps = 1e-06 e intervalo [0, 1]:
     com 512 retângulo a área é aproximadamente = 0.841471
teste 2: para eps = 1e-05 e intervalo [0, 1]:
     com 128 retângulos a área é aproximadamente = 0.841473
teste 3: para eps = 0.0001 e intervalo [0, 1]:
     com 64 retângulos a área é aproximadamente = 0.84148
teste 4: para eps = 0.001 e intervalo [0, 1]:
     com 16 retângulos a área é aproximadamente = 0.841608
Fim dos testes.
```

Roteiro

- **Baixe** o arquivo `area.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.
- **Procure** entender o que a função `main()` faz simulando-a e examinando os resultados produzidos no exemplos acima. Assim, você entenderá como a variável `f_x` faz referência a uma função e o seu uso como argumento de outra função.
- **Implemente**, **teste** e submeta para avaliação a função `area_por_retangulos()`. Você pode alterar a função `main()` para executar apenas os exemplos com `area_por_retangulos()`.
- **Implemente**, **teste** e submeta para avaliação a função `area_aproximada()`.

Se você ainda não viu, veja está simulação aqui para ajudar entender o mecanismo de passagem de uma função como parâmetro.

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.

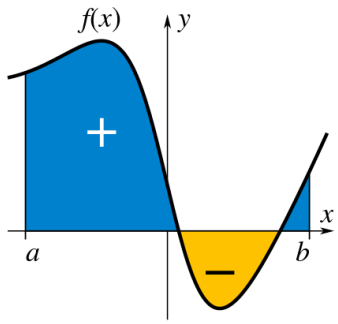
Integrais

Para aquelas e aqueles que desejem entender um pouco sobre a relação entre *área sob funções* e **integrais** do Cálculo I, aqui vai um texto extra. A leitura do que está a seguir não é necessário para fazer o EP.

Dada uma função f de uma variável real x e um intervalo $[a, b]$ da reta real, a **integral definida**

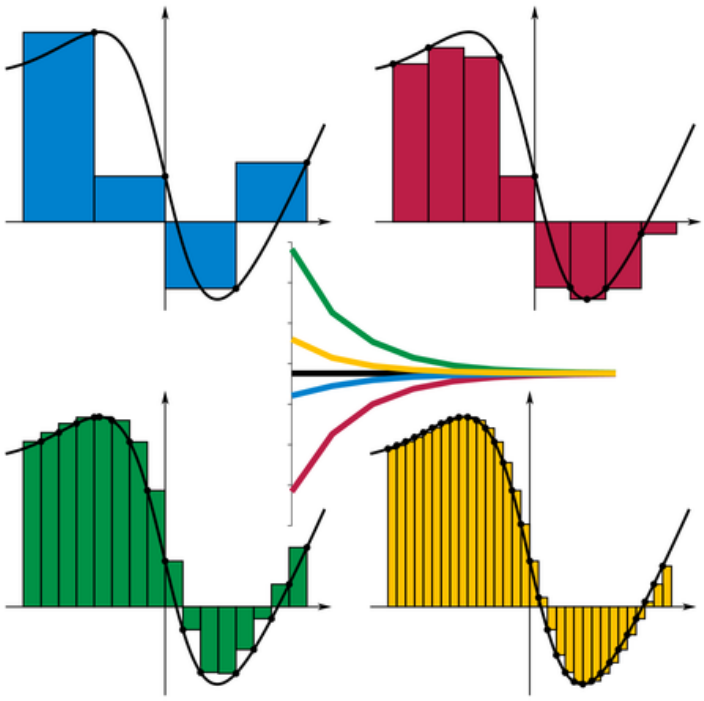
$$\int_a^b f(x)\,dx$$

é definida informalmente como sendo a *área sinalizada* da região do plano limitada pelo gráfico de f, o eixo x (abscissa) as linhas verticais $x = a$ e $x = b$. A área acima do eixo x adiciona ao total enquanto que a área abaixo subtrai do total.

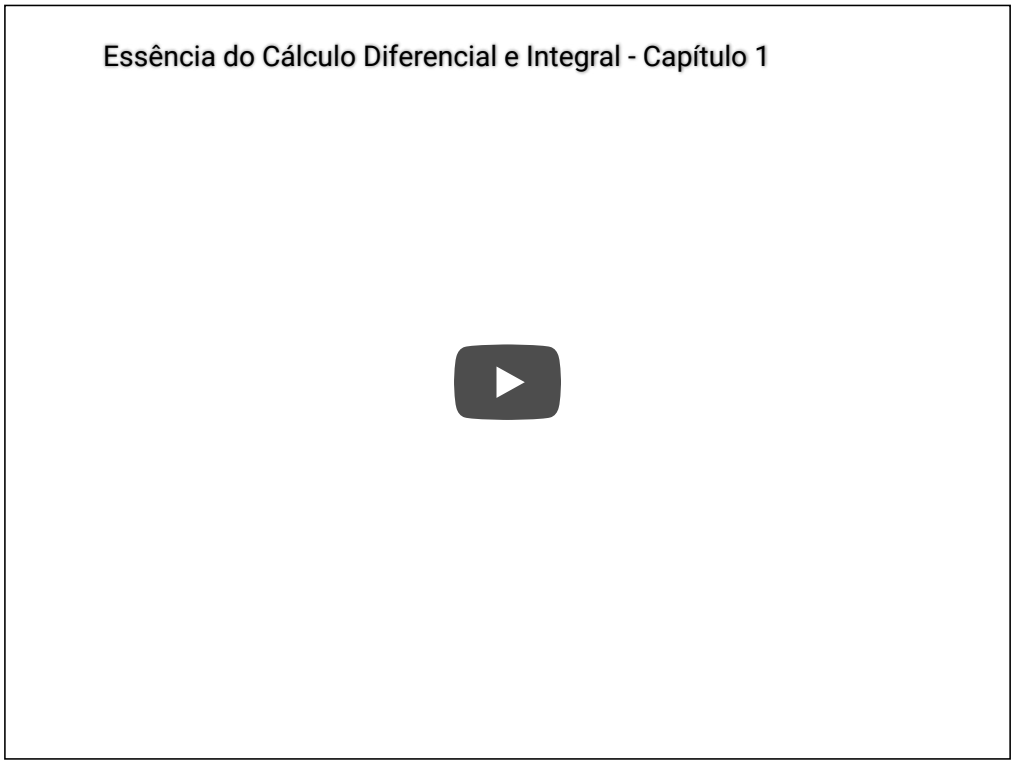


Fonte: Integral (Wikipedia)

Nesse exercício você escreverá uma função que calcula uma *aproximação* do valor da integral definida de uma dada função `f_x()` em um intervalo $[a, b]$. Par isso você utilizará o método dos retângulos e avaliará a qualidade da aproximação através do erro relativo.



Fonte: Riemann sum (Wikipedia)



.

Arquivos requeridos

area.py

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

'''

#---------------------------------------------------------------------
# Constantes que você pode utilizar nesse exercício
# Em notação científica 1.0e-6 é o o mesmo qoe 0.000001 (10 elevado a -6)
EPSILON  = 1.0e-6

#---------------------------------------------------------------------
# O import abaixo permite que o programa utilize todas as funções do módulo math,
# como por exemplo, math.exp e  math.sin.
import math

#---------------------------------------------------------------------
def main():
    '''() -> None

    Modifique essa função, escrevendo outros testes.
    '''
    # escolha a função que desejar e atribuia a f_x
    # f_x = math.cos
    # f_x = math.sin
    # f_x = math.exp # etc, para integração com outras funções.
    # f_x = identidade     # identidade() definidas mais adiante
    # f_x = circunferencia # circunferencia() definida mais adiante
    # f_x = exp            # exp() definida mais adiante

    print("Início dos testes.")
    # Testes da f_x
    nome = f_x.__name__ # nome da f_x usada
    print(f"A função f_x usada nos testes é {nome}()")
    print(f"Valor de f_x(0.0)= {f_x( 0.0 )}")
    print(f"Valor de f_x(0.5)= {f_x( 0.5 )}")
    print(f"Valor de f_x(1.0)= {f_x( 1.0 )}")

    # testes da função área_por_retangulos
    print()
    print("Área por retângulos:")
    a, b = 0, 1 # intervalo [a,b]
    k = 1       # número de retângulos
    n = 3       # número de iterações
    i = 0
    while i < n:
        print(f"teste {i+1}: para {k} retângulos no intervalo [{a}, {b}]:")
        print(f"    área aproximada = {area_por_retangulos(f_x, a, b, k):g}")
        k *= 10
        i += 1

    # testes da função área_aproximada
    print()
    print("Área aproximada:")
    a, b = 0, 1 # intervalo
    k, area = area_aproximada(f_x, a, b) # número de retângulos e aproximação
    print(f"teste 1: para eps = {EPSILON:g} e intervalo [{a}, {b}]:")
```

```
    print(f"teste 1: para eps = {EPSILON:g} e intervalo [{a}, {b}]: ")
    print(f"     com {k} retângulo a área é aproximadamente = {area:g}")
    eps = 1e-6 # erro relativo aceitável
    i = 1
    n = 4
    while i < n:
        eps *= 10 # aumenta o erro relativo aceitável
        k, area = area_aproximada(f_x, a, b, eps)
        print(f"teste {i+1}: para eps = {eps:g} e intervalo [{a}, {b}]:")
        print(f"     com {k} retângulos a área é aproximadamente = {area:g}")
        i += 1

    print("Fim dos testes.")

#---------------------------------------------------------------------
# FUNÇÃO AUXILIAR PARA TESTE: função f(x)=x
def identidade( x ):
    ''' (float) -> float

    RECEBE um valor x.
    RETORNA o valor recebido.
    EXEMPLOS:
    In [6]: identidade(3.14)
    Out[6]: 3.14
    In [7]: identidade(1)
    Out[7]: 1
    In [8]: identidade(-3)
    Out[8]: -3
    '''
    return x

#---------------------------------------------------------------------
# FUNÇÃO AUXILIAR PARA TESTE: função f(x)=sqrt(1 - x*x)
def circunferencia( x ):
    ''' (float) -> float

    RECEBE um valor x.
    RETORNA um valor y >= 0 tal que (x,y) é um ponto na circunferência de raio 1 e centro (0,0).
    PRÉ-CONDIÇÃO: a função supõe que x é um valor tal que -1 <= x <= 1.
    EXEMPLOS:
    In [9]: circunferencia(-1)
    Out[9]: 0.0
    In [10]: circunferencia(0)
    Out[10]: 1.0
    In [11]: circunferencia(1)
    Out[11]: 0.0
    '''
    y = math.sqrt( 1 - x*x )
    return y
#---------------------------------------------------------------------
# FUNÇÃO AUXILIAR PARA TESTE: função f(x) = e^x
def exp( x ):
    ''' (float) -> float
    RECEBE um valor x.
    RETORNA (uma aproximação de) exp(x).
    EXEMPLOS:
    In [12]: exp(1)
    Out[12]: 2.718281828459045
    In [13]: exp(0)
    Out[13]: 1.0
    In [14]: exp(-1)
    Out[14]: 0.36787944117144233
    '''
    y = math.exp( x )
    return y # return math.exp( x )

#---------------------------------------------------------------------
#
def erro_rel(y, x):
    ''' (float, float) -> float

    RECEBE dois números x e y.
    RETORNA o erro relativo entre eles.
    EXEMPLOS:
    In  [1]: erro_rel(0, 0)
    Out [1]: 0.0
    In  [2]: erro_rel(0.01, 0)
    Out [2]: 1.0
    In  [3]: erro_rel(1.01, 1.0)
    Out [3]: 0.01
    '''
    if x == 0 and y == 0:
        return 0.0
    elif x == 0:
        return 1.0
    erro = (y-x)/x
    if erro < 0:
        return -erro
    return erro

#---------------------------------------------------------------------
def area_por_retangulos(f, a, b, k):
    '''(function, float, float, int) -> float
```

```
186
187      RECEBE uma função f, dois números a e b e um inteiro k.
188      RETORNA uma aproximação da área sob a função f no intervalo [a,b]
189          usando k retângulos.
190      PRÉ-CONDIÇÃO: a função supõe que a função f é continua no intervalo [a,b] e que
191           f(x) >= 0 para todo x, a <= x <= b.
192      EXEMPLOS:
193      In [15]area_por_retangulos(identidade, 0, 1, 1)
194      Out[15]: 0.5
195      In [16]:area_por_retangulos(circunferencia, -1, 0, 1)
196      Out[16]: 0.8660254037844386
197      '''
198      # escreva a sua solução a seguir
199      # remova ou modifique a linha abaixo como desejar
200      return 0.0 # aproximação
201
202  #---------------------------------------------------------------------
203  def area_aproximada(f, a, b, eps=EPSILON):
204      '''(function, float, float, float) -> int, float
205
206      RECEBE uma função f, dois números a, b, eps.
207      RETORNA um inteiro k e uma aproximação da área sob a função f no intervalo [a,b]
208          usando k retângulo.
209
210      O valor de k deve ser a __menor potência__ de 2 tal que o erro relativo
211      da aproximação retornada seja menor que eps.
212
213      Assim, os possíveis valores de k são 1, 2, 4, 8, 16, 32, 64, ...
214
215      PRÉ-CONDIÇÃO: a função supõe que a função f é continua no intervalo [a,b] e que
216           f(x) >= 0 para todo x, a <= x <= b.
217
218      EXEMPLOS:
219      In [22]: area_aproximada(identidade, 1, 2)
220      Out[22]: (2, 1.5)
221
222      In [23]: area_aproximada(exp, 1, 2, 16)
223      Out[23]: (2, 4.6224728167337865)
224      '''
225      # escreva o corpo da função
226      # remova ou modifique a linha abaixo como desejar
227      return 1, 0.0  # para retornar um int e um float
228                     # basta separá-los por vírgula
229
230  ###########################################################
231  ###                  FIM                                ###
232  ###########################################################
233  #
234  #  NÃO MODIFIQUE AS LINHAS ABAIXO
235  #
236  # Esse if serve para executar a função main() apenas quando
237  # este é o módulo a partir do qual a execução foi iniciada.
238  if __name__ == '__main__':
239      main()
240
```

VPL

◄ EP08

Seguir para...

EP06 ►

2012
AACCs/FFLCH
Pró-Reitoria de Pós-Graduação
Outros
Suporte »
Documentação
HelpDesk e Contato
Guia de uso
Sobre
Português - Brasil (pt_br)
Deutsch (de)
English (en)
Español - Internacional (es)
Français (fr)
Português - Brasil (pt_br)