15 Reunião 15: 20/OUT/2020

15.1 Reuniões passadas

- tipos: `int`, `str`, `float`, `bool`, `NoneType`
- comandos de seleção
- comando de repetição `while`
- funções:
  - criar uma nova função permite dar nome a um conjunto de comandos
  - funções diminuem o tamanho de um programa através da eliminação de repetições
  - dividir um programa em funções facilita o teste dos componentes
  - funções bem projetadas podem ser usadas por vários programas

Figure 1: Fonte: *WALL-E*, Disney-Pixar

15.2 Exercício: séries de potências

Em matemática, uma série é essencialmente uma descrição da operação de adicionar infinitas quantidades, uma após a outra. O estudo de séries é uma parte importante do cálculo e análise matemática.

(a) Escreva uma função `modulo()` que **recebe** um valor `x` e retorna o valor absoluto de `x`.

(b) Escreva uma função `fatorial()` que **recebe** um número inteiro `n` e retorna o fatorial de `n`.

(c) Escreva uma função `E()` que **recebe** um número real `x` e um número real `epsilon > 0` e calcular uma `epsilon`-aproximação do valor `Sx` da série:

$$Sx = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$

Em uma `epsilon`-aproximação são adcionados à soma todos os termos até o primeiro de *valor absoluto* menor do que `epsilon`.

(d) Escreve uma função `main()` que lê uma número `x` e um número real `epsilon > 0` e imprime uma `epsilon`-aproximação de S_x.

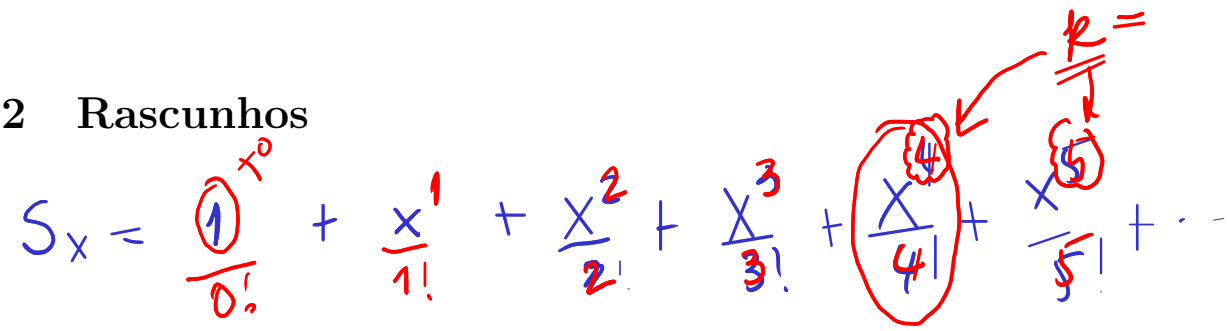
15.2.1 Exemplos

Os exemplos a seguir mostram os cálculos intermediários.

```
Digite o valor de x: 1
Digite o valor de eps: 0.1
k= 0, termo(0)=1.000000, Sx=1.000000
k= 1, termo(1)=1.000000, Sx=2.000000
k= 2, termo(2)=0.500000, Sx=2.500000
k= 3, termo(3)=0.166667, Sx=2.666667
k= 4, termo(4)=0.041667, Sx=2.708333
E(1.000000) = 2.708333
```

```
Digite o valor de x: 1
Digite o valor de eps: 0.01
k= 0, termo(0)=1.000000, Sx=1.000000
k= 1, termo(1)=1.000000, Sx=2.000000
k= 2, termo(2)=0.500000, Sx=2.500000
k= 3, termo(3)=0.166667, Sx=2.666667
k= 4, termo(4)=0.041667, Sx=2.708333
k= 5, termo(5)=0.008333, Sx=2.716667
E(1.000000) = 2.716667
```

15.2.2 Rascunhos



15.2.3 Solução

(a) `modulo()` **recebe** um valor `x`
e **retorna** o valor absoluto de `x`.

```
def modulo(x):
    '''(número) -> número
       Retorna o valor absoluto de x
    '''
    if x < 0:
        return -x # abandona a função aqui
    return x # só chega nesta linha se x >= 0
```

(b) `fatorial()` **recebe** um número inteiro `n`
e retorna `n!`.

```
def fatorial(n):
    '''(int) -> int
       Retorna n!
    '''
    nfat = 1
    i = 2
    while i <= n:
        nfat *= i
        i += 1
    return nfat
```

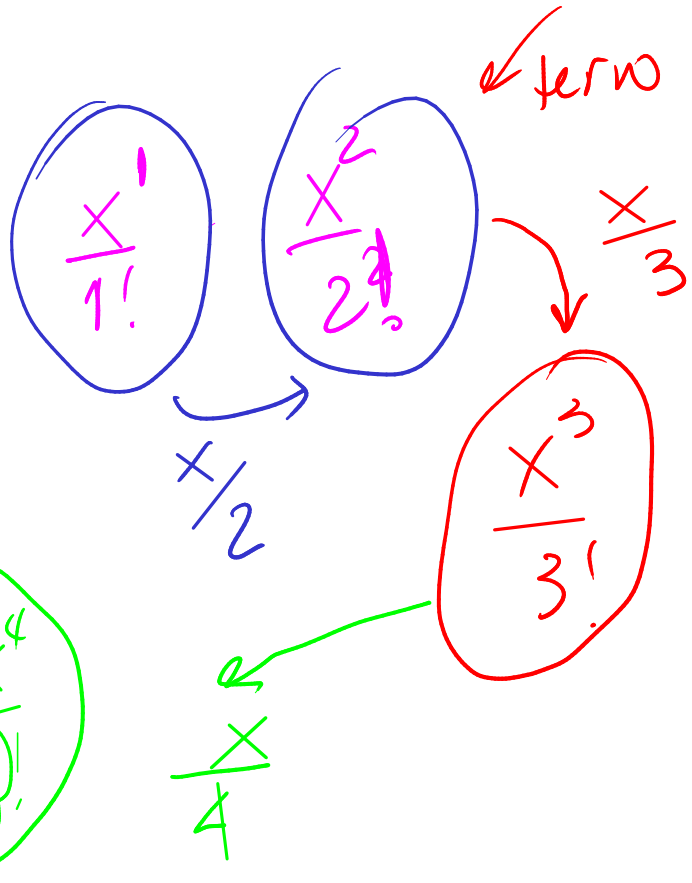
(c) `E()` que **recebe** um número real `x` e um
número real `epsilon > 0` e calcular uma `epsilon`-aproximação
do valor `Sx` da série:

$$\texttt{Sx} = 1 + \texttt{x} + \frac{\texttt{x}^2}{2!} + \frac{\texttt{x}^3}{3!} + \ldots + \frac{\texttt{x}^k}{\texttt{k}!} + \ldots$$

Em uma `epsilon`-aproximação são adicionados à soma todos
os termos até o primeiro de *valor absoluto* menor do que `epsilon`.

$$Sx = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^k}{k!} + \dots$$

$$Sx = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^k}{k!} + \dots$$



$$Sx = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$

$$Sx = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$

```
def E(x, eps):
    '''(float, float) -> float

    Recebe números reais x e eps e retorna uma eps-aproximação d
    '''
    # inicialização das variáveis
    Sx    = 1
    termo = 1
    k     = 0
    print(f"k={k}, termo({k})={termo} Sx={Sx}\n")

    while modulo(termo) > eps:
        k = k + 1

        # calcule k!
        kfat = fatorial(k)
        # calcule x elevado a k
        xk = x**k
        # calcule o proximo termo
        termo = xk / kfat
        # incremente o termo
        Sx = Sx + termo

        print(f"k={k} termo({k})={termo} Sx={Sx}")

    return Sx
```

(d) `main()` que **lê** uma número `x` e um número real `epsilon > 0` e imprime uma `epsilon`-aproximação de `S_x`.

```
def main():
    # leia x e eps
    x   = float(input("Digite o valor de x: "))
    eps = float(input("Digite o valor de eps: "))
    print(f"E({x})={E(x,eps)"))
```

(d) `main()` que **lê** uma número `x` e um número real `epsilon > 0` e imprime uma `epsilon`-aproximação de `S_x`.

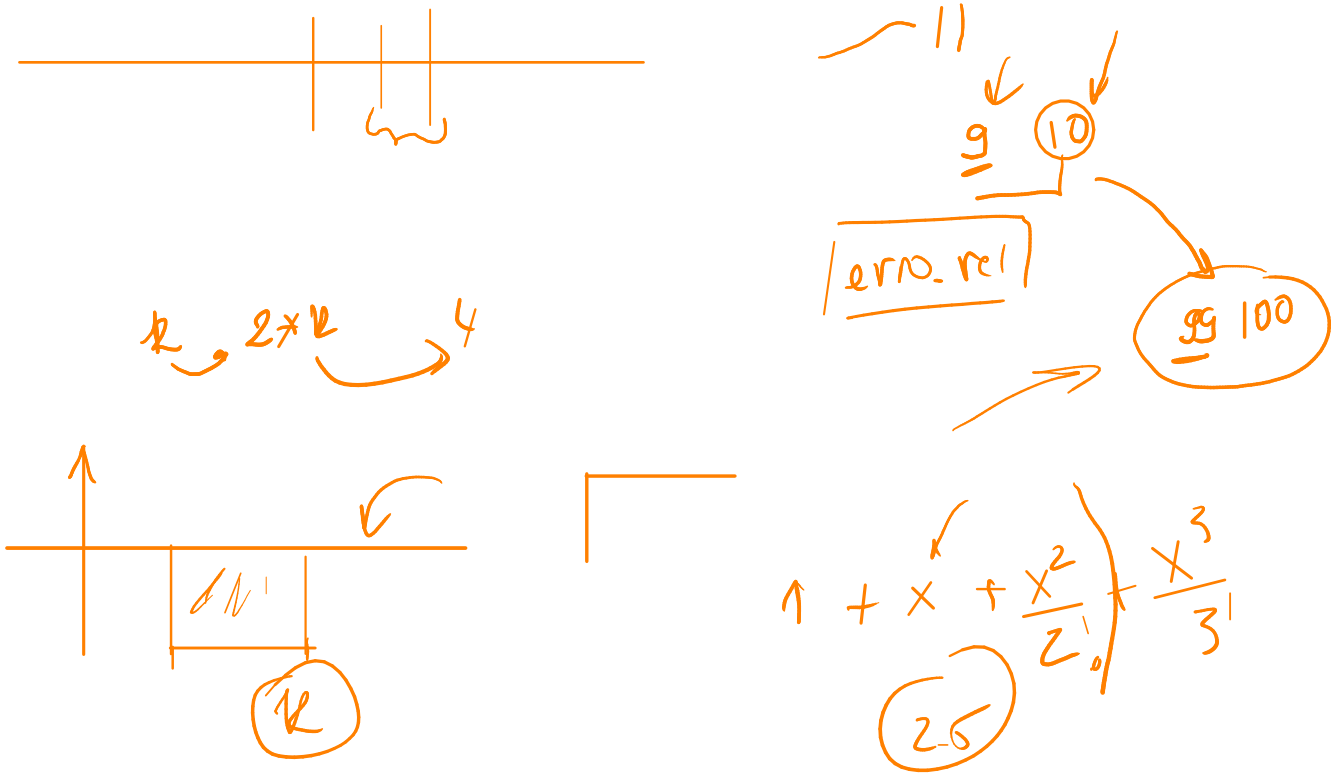
```
import math

def main():
    # leia x e eps
    x   = float(input("Digite o valor de x: "))
    eps = float(input("Digite o valor de eps: "))
    print(f"E({x})={E(x,eps)} math.exp({x})={math.exp(x)}")
```

15.2.4 Exemplos

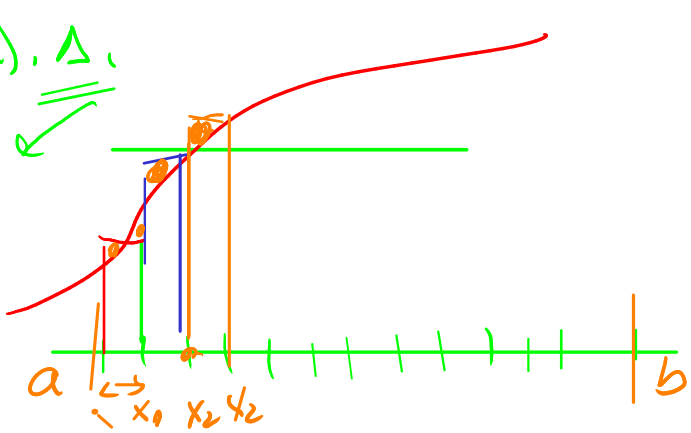
Os exemplos a seguir mostram os cálculos intermediários.

```
Digite o valor de x: 1
Digite o valor de eps: 0.1
k= 0, termo(0)=1.000000, Sx=1.000000
k= 1, termo(1)=1.000000, Sx=2.000000
k= 2, termo(2)=0.500000, Sx=2.500000
k= 3, termo(3)=0.166667, Sx=2.666667
k= 4, termo(4)=0.041667, Sx=2.708333
E(1.000000) = 2.708333

Digite o valor de x: 1
Digite o valor de eps: 0.01
k= 0, termo(0)=1.000000, Sx=1.000000
k= 1, termo(1)=1.000000, Sx=2.000000
k= 2, termo(2)=0.500000, Sx=2.500000
k= 3, termo(3)=0.166667, Sx=2.666667
k= 4, termo(4)=0.041667, Sx=2.708333
k= 5, termo(5)=0.008333, Sx=2.716667
E(1.000000) = 2.716667
```

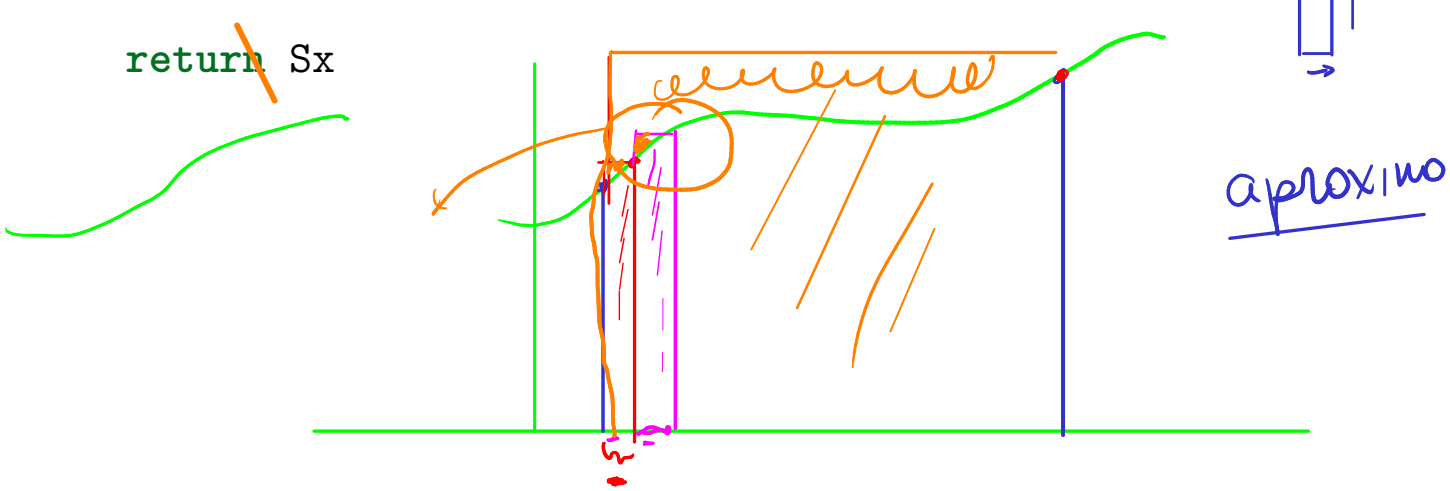
$$Sx = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2}! + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$



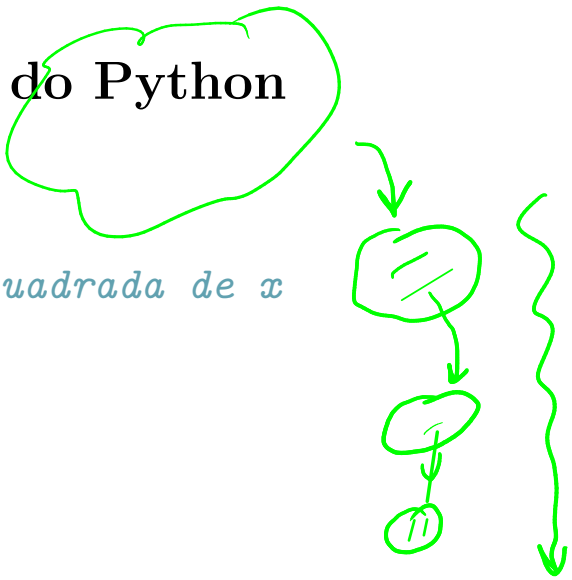
$$Sx = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2}! + \frac{x^3}{3!} + \ldots + \frac{x^k}{k!} + \ldots$$




```
def E(x, eps):
    '''(float, float) -> float

    Recebe números reais x e eps e retorna uma eps-aproximação d
    '''
    # inicialização das variáveis
    Sx    = 1
    termo = 1
    k     = 0
    print(f"k={k}, termo({k})={termo} Sx={Sx}\n")

    while modulo(termo) > eps:
        k = k + 1

        # calcule o proximo termo
        termo = termo * x / k

        # incremente o termo
        Sx = Sx + termo

        print(f"k={k} termo({k})={termo} Sx={Sx}")

    return Sx
```



15.3 Biblioteca matemática do Python



```
    import math

    math.sqrt(x) # retorna raiz quadrada de x
    math.fabs(x) # retorna |x|
    math.sin(x)  # seno(x)
    math.cos(x)  # cosseno(x)
    math.exp(x)
```

```
In [1]: import math

In [2]: help(math)
Help on module math:

NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.7/library/math

    The following documentation is automatically generated from t
    source files.  It may be incomplete, incorrect or include fea
    are considered implementation detail and may vary between Pyt
    implementations.  When in doubt, consult the module reference
    location listed above.

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x)
```

```
        Return the arc cosine (measured in radians) of x.

    atan2(y, x)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considere

    cos(x)
        Return the cosine of x (measured in radians).


    pow(x, y)
        Return x**y (x to the power of y).

    sin(x)
        Return the sine of x (measured in radians).

    sqrt(x)
        Return the square root of x.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586
```