Paradigmas de Linguagens de Programação em Python

Fundamentos dos Subprogramas

Prof. Dr. Daniel Caetano

2022 - 1

Compreendendo o problema

- **Situação:** Desenvolver um ERP...
  - Contabilidade, RH, Suprimentos, Patrimônio....

  - Dá pra fazer isso em um único bloco de código

Mentimeter

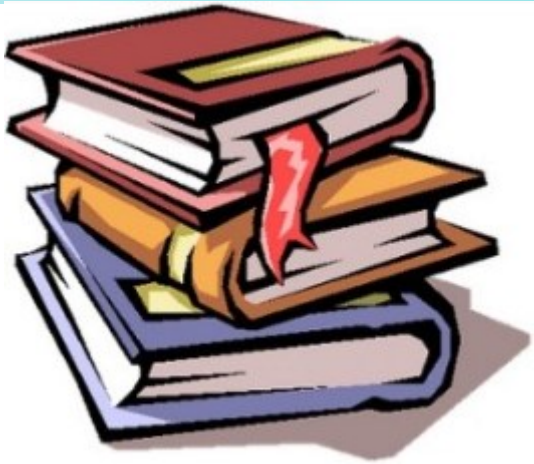
https://www.menti.com/

Objetivos

- Compreender os Módulos do Python
- Compreender as funções no Python
- Reforçar a ideia de escopos em Python
- Capacitar para a criação de funções em Python

- **Estudar para a prova!**

Bibliografia da Aula



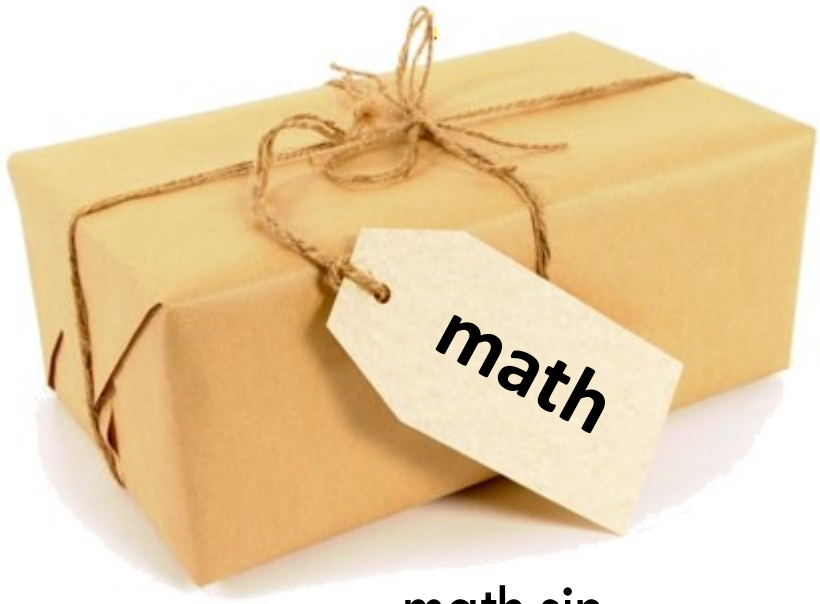
Material	Acesso ao Material
Apresentação	https://www.caetano.eng.br/aulas/2022a/ara0066.php (Paradigmas de Programação – Aula 11)
Livro Texto	Capítulo 9, páginas 364 a 372
Aprenda Mais!	• Texto: Princípios de Programação Funcional em Python https://wiki.python.org.br/PrincipiosFuncionais • Vídeo: Recusão. https://www.youtube.com/watch?v=KEEKn7Me-m • Vídeo: Como usar funções no Python https://www.youtube.com/watch?v=NSbOtYzIQI0

Módulos

https://www.menti.com/

Módulos em Python

- Módulos são conjunt
  - "Pacotes"



- Já os usamos: import
  - math
  - string

math.sin
math.log
...

- Podemos criar os nossos...!
  - Mas precisamos aprender a criar funções antes!

SUBPROGRAMAS:
FUNÇÕES SIMPLES

Funções Simples

- Situação: imprimir 5x o seguinte texto:

Sistema de Impressão v.1.0, (c) Daniel Caetano

- Um jeito de fazer seria usar vários “print”:

aula10ex01.py

```
# Imprime 5x uma mensagem
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
print (“Sistema de Impressão v.1.0, (c) Daniel Caetano”)
```

E se eu quiser mudar a versão?

Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula10ex01a.py

```
# Imprime 5x uma mensagem
def mostra_mensagem():
    print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
```

Indentação

Vamos experimentar?

Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula10ex01a.py

```
# Imprime 5x uma mensagem
def mostra_mensagem():
        print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")

mostra_mensagem()
```

"chamar" a função:
Solicitar ao computador que a execute

Observe o uso dos **parênteses**!

Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula10ex01a.py

```
# Imprime 5x uma mensagem
def mostra_mensagem():
        print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
```

E para mudar a versão?

```
# Repita quantas vezes quiser!
```

Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula10ex01a.py

FUNÇÃO

```
# Imprime 5x uma mensagem
def mostra_mensagem():
        print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()          # I
```

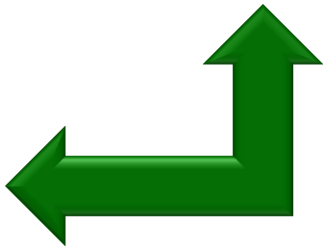
Declaração da Função: define o **nome** e o **código** da função

Funções Simples

- Será que não tem um jeito mais simples?
  - Há vários!
- Um deles: definir uma função

aula10ex01a.py

```
# Imprime 5x uma mensagem
def mostra_mensagem():
        print ("Sistema de Impressão v.1.0, (c) Daniel Caetano")
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
mostra_mensagem()
```



As funções devem ser declaradas antes de serem chamadas!

Outro Exemplo de Função

- Crie a função para a assinatura do e-mail:

  Atenciosamente,

  Prof. Daniel Caetano

  prof@caetano.eng.br

aula10ex02.py

```
# Imprime a assinatura do e-mail
def assinar():
        print (“Atenciosamente,”)
        print (“Prof. Daniel Caetano”)
        print (“prof@caetano.eng.br”)

assinar()
```

Experimentemos!

Nosso Primeiro Módulo

Criando um Módulo

- Crie o arquivo abaixo

mensagem.py — Nome do Módulo

```
# Imprime a assinatura do e-mail
def assinar():          # Nome da Função
        print ("Atenciosamente,")
        print ("Prof. Daniel Caetano")
        print ("prof@caetano.eng.br")
```

Usando um Módulo

- Crie o programa abaixo e execute

aula10ex03.py

```
import mensagem
# Usa o módulo "mensagem.py"
print ("Programa de demonstração do uso de um módulo")

mensagem.assinar()
```

Tenho que escrever tudo isso?

Apelido de um Módulo

- Crie o programa abaixo e execute

aula10ex03.py

```
import mensagem as msg
# Usa o módulo "mensagem.py"
print ("Programa de demonstração do uso de um módulo")

msg.assinar()
```

Apelido do Módulo

Módulos Mais Úteis

- Como tornar nossos módulos mais úteis?
  - Criando funções mais úteis!

- Como criar funções mais úteis?

Recordando:

Escopo de Variáveis

Escopo de Variáveis

- Variáveis diferentes podem ter mesmo nome
  - Seu professor chama “Daniel”
  - Com certeza você conhece mais algum “Daniel”
- Como diferenciar?
  - Contexto (ou Escopo)!
    - Na faculdade, “Daniel” é o professor
    - Em sua casa, “Daniel” pode ser seu irmão
- Em Python, qual o “escopo”?
  - Escopo principal (ou global, fora das funções)
  - Escopo da função (ou local)

Exemplo de Escopo de Variáveis

- Vamos ver escopos na prática

aula10ex04.py

```
# Função
def aniversário():
        idade = idade + 1

# Programa principal
idade = 10
print("A idade antes do aniversário:", idade)
aniversário(idade)
print("A idade depois do aniversário:", idade)
```

Escopo da função aniversário (local)

Escopo global

Funcionou?

SUBPROGRAMAS:
FUNÇÕES COM PARÂMETROS

Funções com Parâmetro

- Até agora, funções funcionam sempre igual!
  - Por exemplo: assinatura de um único professor

Atenciosamente,

Prof. Daniel Caetano

assinar()

- Como fazer função de assinatura genérica?

Atenciosamente,

Prof. *[nome]*

assinar("Daniel Caetano")

Funções com Parâmetro

- Crie a função para a assinatura do e-mail:

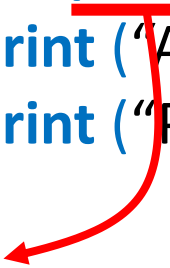
  Atenciosamente,

  Prof. *[nome]*

aula10ex05.py

```
# Imprime a assinatura do e-mail genérica
def assinar(nome):
        print ("Atenciosamente,")
        print ("Prof.", nome)

assinar()
```



Funcionou?

Funções com Parâmetro

- Crie a função para a assinatura do e-mail:

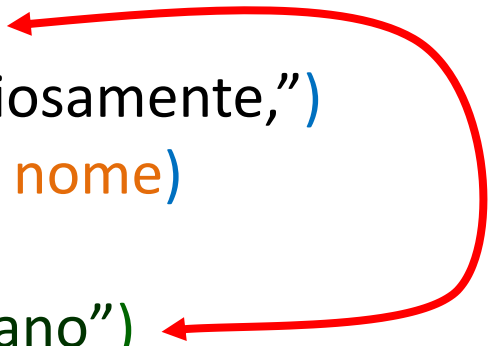
  Atenciosamente,

  Prof. *[nome]*

aula10ex05.py

```
# Imprime a assinatura do e-mail genérica
def assinar(nome):
        print ("Atenciosamente,")
        print ("Prof.", nome)

assinar("Daniel Caetano")
```



E agora?

Funções com Vários Parâmetros

- Crie a função para a assinatura completa:

  Atenciosamente,

  Prof. *[nome]*

  [e-mail]

aula10ex06.py

```
# Imprime a assinatura do e-mail genérica completa
def assinar(nome, email):
        print ("Atenciosamente,")
        print ("Prof.", nome)
        print (email)

assinar("Daniel Caetano", "prof@caetano.eng.br")
```

Funcionou?

Funções com Parâmetros

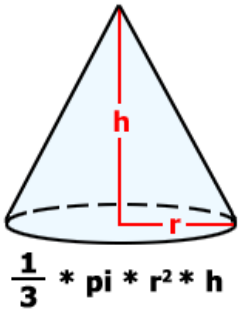
- Função para calcular IMC = P/A^2

aula10ex07.py

```
# Calcula e imprime IMC
def imc(p, a):
        imc = p/a**2
        print("O IMC é:", imc)

# Testa função
imc(50, 1.60)
imc(75, 1.70)
imc(85, 1.75)
```

Funções com Parâmetros



- Calcular o volume de um cone

aula10ex08.py

```
# Imprime volume do cone
def volume_cone(r, h):
        v = (1/3) * 3.1415 * r**2 * h
        print("O colume do cone é: %.2f" %(v))

# Testa função
volume_cone(10, 2)
volume_cone(5, 3)
volume_cone(2.5, 10.2)
```

Ou seja: criar uma função é ensinar uma tarefa nova ao computador... E dar um **nome** para ela!

SUBPROGRAMAS:
FUNÇÕES COM RETORNO

Funções com Retorno

- Vimos: fazer conta e mostrar o resultado
  - Posso usar o resultado depois?
  - Vamos testar?

aula10ex09.py

```
# Calcula área do círculo
def área_circulo(r):
        área = 3.141592 * r**2

# Testa função
área_circulo(2)
print("A área do círculo é:", área)
```

Funcionou?

As variáveis que existem na função são diferentes das que existem no programa principal!

Funções com Retorno

- Vimos: fazer conta e mostrar o resultado
  - Posso usar o resultado depois?
  - Como resolver?

aula10ex09.py

```
# Calcula área do círculo
def área_circulo(r):
        área = 3.141592 * r**2
        return área

# Testa função
a = área_circulo(2)
print("A área do círculo é:", a)
```

Funcionou?

Return serve para "devolver" um valor de uma função para o programa principal

Funções x Procedimentos

- Subprogramas retornam resultados... ou não.
  - Alguns autores dão nomes diferentes

- Procedimentos (procedures)
  - Quando **não retorna** um valor
- Funções (functions)
  - Quando **retorna** um valor

print("...")

int("...")
input("...")
math.sin(n)

Exemplo de Execução:

Subprogramas com Retorno de Valores

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
    pnome = input("Por favor, digite seu primeiro nome: ")
    snome = input("Por favor, digite seu sobrenome: ")
    nome = pnome + " " + snome
    return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
    pnome = input("Por favor, digite seu primeiro nome: ")
    snome = input("Por favor, digite seu sobrenome: ")
    nome = pnome + " " + snome
    return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

Subprogramas com Retorno

- Vejamos como funciona o programa abaixo:

aula10ex10.py

Experimente no http://pythontutor.com/visualize.html

```
# Função que lê nome do usuário
def leitura_de_nome():
        pnome = input("Por favor, digite seu primeiro nome: ")
        snome = input("Por favor, digite seu sobrenome: ")
        nome = pnome + " " + snome
        return nome

# Programa de Boas Vindas
print("Programa Exemplo com Funções")
print("============================")
usuario = leitura_de_nome ()
print("Bom dia,", usuario, "!")
```

RECURSÃO

https://www.menti.com/

Recursão

- Forma de implementação em que...
  - Executa a si própria em determinadas situações

```
def minhaFuncao()
        print("Ah!)
        minhaFuncao()
```

- Algumas coisas são mais simples assim
  - Mas usualmente gastam mais memória!
- Exemplo?

Cálculo de Fatorial – Sem recursão

aula10ex11.py

```
# Cálculo de Fatorial sem Recursão
print("Programa Fatorial sem Recursão")
print("============================")
n = int( input("Digite um inteiro não negativo: ")
fat = 1
for i in range (1, n+1):
        fat = fat * i
print("O fatorial de", n, "é",  fat)
```

Cálculo de Fatorial – Com recursão

aula10ex12.py

```
# Cálculo de Fatorial com Recursão
def fatorial(n):
        if n == 1:
                return n
        else:
                return n * fatorial(n-1)

print("Programa Fatorial com Recursão")
print("============================")
n = int( input("Digite um inteiro não negativo: ")
fat = fatorial(n)
print("O fatorial de", n, "é",  fat)
```

ATIVIDADE AVALIATIVA E

Atividade 1

- Faça função que calcule a área do trapézio, dados:
  - Base maior
  - Base menor
  - Altura
- Lembrando que a área pode ser calculada por:

$$area = \frac{(bmaior + bmenor).altura}{2}$$

- O programa principal deve pedir os valores e usar a função para calcular a área

Atividade 2

- Faça um programa em Python com uma função chamada **somaImposto**. A função possui dois parâmetros formais:
  - **taxaImposto**, que é a quantia de imposto sobre vendas expressa em porcentagem e
  - **custo**, que é o custo de um item antes do imposto. A função "altera" o valor de custo para incluir o imposto sobre vendas.
- O programa principal deve pedir os dados e usar a função para calcular preço final de um produto.

Atividade 3

- Faça um programa que converta da notação de 24 horas para a notação de 12 horas. Por exemplo, o programa deve converter 14:25 em 2:25 P.M.
- A entrada é dada em dois inteiros.
- Deve haver pelo menos duas funções: uma para fazer a conversão e uma para a saída.
- Registre a informação A.M./P.M. como um valor "A" para A.M. e "P" para P.M. Assim, a função para efetuar as conversões terá um parâmetro formal para registrar se é A.M. ou P.M.
- Inclua um loop que permita que o usuário repita esse cálculo para novos valores de entrada todas as vezes que desejar, digitando um valor negativo para a hora quando quiser encerrar.

ENCERRAMENTO

Resumo e Próximos Passos

- Módulos e funções: reaproveitar de código
- As variáveis possuem um escopo
- Funções:
  - Recebem parâmetros / retornam resultados
- **Pós Aula**: Saiba Mais, A Seguir... e Desafio!
  - No mural: https://padlet.com/djcaetano/paradigmas

- Avaliações e encerramento!
  - Não esqueçam do crédito digital!

PERGUNTAS?