F<BR/C>
DE SOFTWARE

UC: PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS

PROFESSOR: THIAGO ALMEIDA

FUNÇÕES

- No contexto da programação, podemos definir uma função como sendo um bloco de código que pode receber argumentos de entrada, realizar procedimentos específicos e retornar (ou não) um valor.
- O bloco de código que define uma função possui um certo isolamento em relação ao restante do código, e pode ser convocada (chamada) a executar um número arbitrário de vezes em um programa.
- A partir do momento que começamos a aprender o Python, já utilizamos funções prontas, como o **print(), type(), input(), float(), len(), range(), sum();**
  - Funções utilizadas em listas:
  - **lista.append(); lista.pop(); lista.remove(); lista.sort();**

FUNÇÕES

- Uma função é um bloco de código que só é executado quando é chamado.
- Você pode passar dados, conhecidos como parâmetros, para uma função.
- Uma função pode retornar dados como resultado.
- Funções são recursos poderosos das linguagens de programação. Ao desenvolver uma aplicação utilizamos elas a todo momento, quer sejam as que nós mesmos escrevemos para as rotinas especificas ou aquelas já disponibilizadas pela linguagem de programação escolhida, suas bibliotecas e frameworks.

VANTAGENS DE UTILIZAR FUNÇÕES

- Permitem a reutilização de código;
- Possibilitam a decomposição de procedimentos;
- Podem tornar um código mais legível;
- Facilitam a manutenção e a correção de erros;
- Facilitam a manutenção e a correção de erros

Exemplo:

```
>>> round ( 6 . 8 )
>>> 7
```

DEF – DEFININDO UMA FUNÇÃO

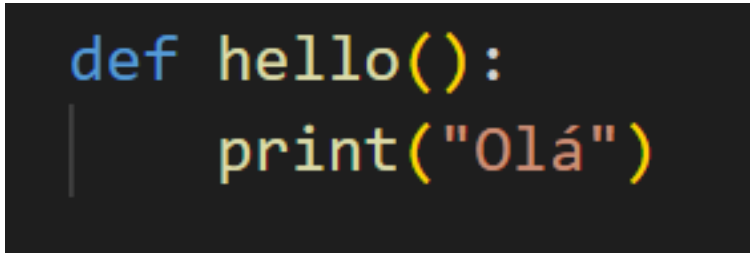
- A sintaxe de uma função é definida por três partes: nome, argumentos e corpo, o qual agrupa uma sequência de linhas que representa algum comportamento.
- No código abaixo, temos um exemplo de **declaração de função em Python**:
- Podemos declarar funções através da cláusula def + nome da função + ():

Exemplo:

```
def  hello ( ):
   print(“ Olá!!! “)
```

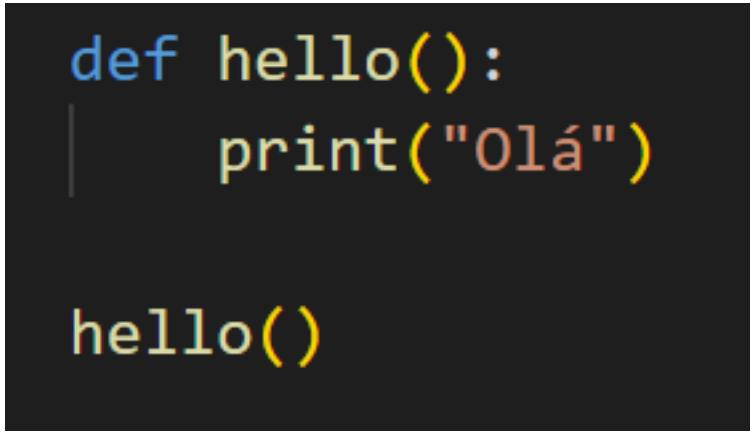
DEF – DEFININDO UMA FUNÇÃO

- As funções iniciam com a declaração **def**, seguida do nome da função, e entre parênteses os parâmetros, caso existam e por último **:** (Dois Pontos) que indica que o código será indentado nas linhas abaixo. Depois de declarada a função, basta passar o corpo contendo o objetivo da função.



CHAMADA DE FUNÇÃO

- Na programação estruturada, o Python roda da primeira até a última linha. Linha por linha. Na ordem.
- No exemplo de código anterior, não.
  É definido uma função, tem um print, mas nada ocorre.
- Só depois que chamamos a função (**hello()**), é que acontece algo. Ou seja, o Python **pula** esses **def**, não roda ele de cara.
  Aliás, se você não tivesse chamado a função via **hello()**, aquele código nem seria executado.



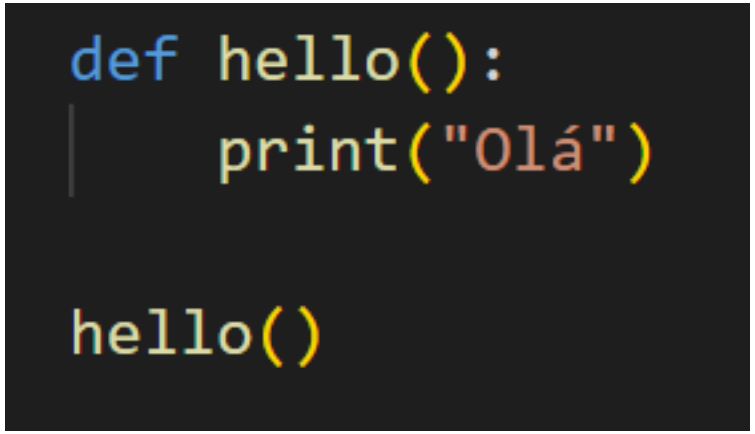
CHAMADA DE FUNÇÃO

- Após definir a função precisamos chamar para executa-lá.

Exemplo:

#definindo função

```
def  hello ( ):
   print(“ Olá!!! “)
#chamando para execução
hello()
```



CHAMADA DE FUNÇÃO

- Quando chamamos uma função de nome **nome()**, o Python sai buscando no código uma **def nome()**, ou seja, ele sabe que você chamou uma função.
- Então ele para tudo e vai buscar onde essa função foi declarada e executa ela.
- Pode haver 1 milhão de linhas de código com funções. Se elas não forem chamadas, simplesmente não executam.

```
def hello():
    print("Olá")

hello()
```

ARGUMENTOS DE FUNÇÃO

- Essa função, de nome hello, tem como objetivo imprimir apenas ***olá!!!***, porém agora iremos passar um argumento (também chamado de parâmetro).
- A palavra reservada def, na primeira linha, explicita a definição da função naquele ponto. Em seguida, entre parênteses, teremos o parâmetro **nome** que receberá uma string.

```
def hello(nome):
    print(f"Olá {nome}")

hello("João")
```

ARGUMENTOS DE FUNÇÃO

- Caso seja necessário, também é possível definir funções com vários argumentos, como no código abaixo:
- Agora, ao invocar essa função, também é necessário informar o segundo parâmetro, que representa a idade que será impressa após o nome.

```
def hello (nome, idade):
    print(f"Olá {nome} sua idade é: {idade}")


hello("João",18)
```

RETORNO DE FUNÇÃO

- Assim como podem receber valores de entrada, as funções também podem produzir valores de saída, provenientes de determinadas operações.
- Nos exemplos anteriores, apenas imprimimos um valor com a função print, sem retornar explicitamente um resultado.
- Se quisermos utilizar ou armazenar o resultado de uma função?
- Para conseguirmos este comportamento, precisamos que nossa função retorne o valor calculado por ela.
- Para isso utilizamos o comando **return**

```
def soma(num1, num2):
    soma = num1 + num2
    return soma

x = soma(42,58)
```

EXERCÍCIOS FUNÇÕES

1 - Crie uma função que necessite de três argumentos, e que imprima o produto desses três argumentos.

EXERCÍCIOS FUNÇÕES

2 – Crie uma função para calcular a exponenciação, que necessite dois argumentos e apresente como resultado a potência. Sendo base elevado ao expoente.

EXERCÍCIOS FUNÇÕES

3 – Crie uma função que imprima a quantidade de dígitos de um determinado número inteiro informado.

EXERCÍCIOS FUNÇÕES

4 – Escreva um programa, com uma função que necessite de um argumento. A função retornar o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.

EXERCÍCIOS FUNÇÕES

5 – Escreva um programa 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. Por fim deve retornar o novo valor para o usuário considerando o percentual do imposto.

EXERCÍCIOS FUNÇÕES

6 – Um comerciante possui uma loja de artigos de R$ 1,99. Para agilizar o cálculo de quanto cada cliente deve pagar ele desenvolveu uma tabela que contém o número de itens que o cliente comprou e ao lado o valor da conta. Desta forma a atendente do caixa precisa apenas contar quantos itens o cliente está levando e olhar na tabela de preços. Você foi contratado para desenvolver uma função que monta esta tabela de preços, que conterá os preços de 1 até 50 produtos, conforme o exemplo abaixo:

```
Lojas Quase Dois - Tabela de preços
1 - R$ 1.99
2 - R$ 3.98
...
50 - R$ 99.50
```

EXERCÍCIOS FUNÇÕES

7 – Crie uma função que efetue o cálculo do salário e como retorno teremos o valor a ser pago conforme o número de horas trabalhadas. Considere a carga horária de 40h por semana como salário base, caso ultrapasse o limite de 40h, o funcionário deve receber 50% a mais por hora excedente.

EXERCÍCIOS FUNÇÕES

8 – Um pescador, comprou um PC para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do MS (50 Kg) deve pagar uma multa de R$ 4,00 por quilo excedente. O pescador precisa que você crie uma função para ler a quantidade de peixes e calcular o excesso. Gravar na variável excesso a quantidade de quilos além do limite e na variável multa o valor da multa que o pescador deverá pagar. Imprima os dados do programa com as mensagens adequadas.

EXERCÍCIOS FUNÇÕES

9 – O gestor de uma rede de shoppings, precisa contratar um sistema para administrar o estacionamento, a principal função do sistema é calcularTempo(). Considere o valor mínimo de R$9,00 por hora e R$ 1,50 por hora adicional. O principal argumento da função é o tempo utilizado em minutos, a função deve **retornar** o valor total. Caso o usuário fique no estacionamento por menos de 15 minutos, o tempo utilizado não será cobrado.

EXERCÍCIOS FUNÇÕES

9 – Adicione na função calcularTempo() do sistema para estacionamento o valor dos impostos pago pelo cliente. Considere o PIS: 0,33% , COFINS: 0,20% e ICMS: 17% no valor e **imprima** o recibo do cliente de acordo com a saída abaixo:

```
Tempo 4.0 horas
PIS R$ 0.45
COFINS R$ 0.27
ICMS R$ 2.30
IMPOSTOS R$ 3.01
TOTAL R$ 13.50
```

EXERCÍCIOS FUNÇÕES

10 – Uma pessoa está interessada em comprar um carro e deseja fazer um financiamento. Ela tem uma quantia X para dar de entrada, uma taxa de juros é definida pelo banco e a pessoa pode escolher o número de parcelas que deseja financiar.

Crie uma função que simule um financiamento, levando em consideração o regime de juros compostos. O programa deve solicitar ao usuário o valor do veiculo, o valor da entrada, a taxa de juros e a quantidade de parcelas. Além disso, o programa deve exibir o valor total pago, a quantia dos juros pagos e o valor de cada parcela. O programa deve apresentar as informações de forma clara e objetiva, facilitando a compreensão por parte do usuário.