

# COM220

## Computação

### Orientada a Objetos I

Aula 01 – Introdução a Linguagem Python

# Conteúdo



2

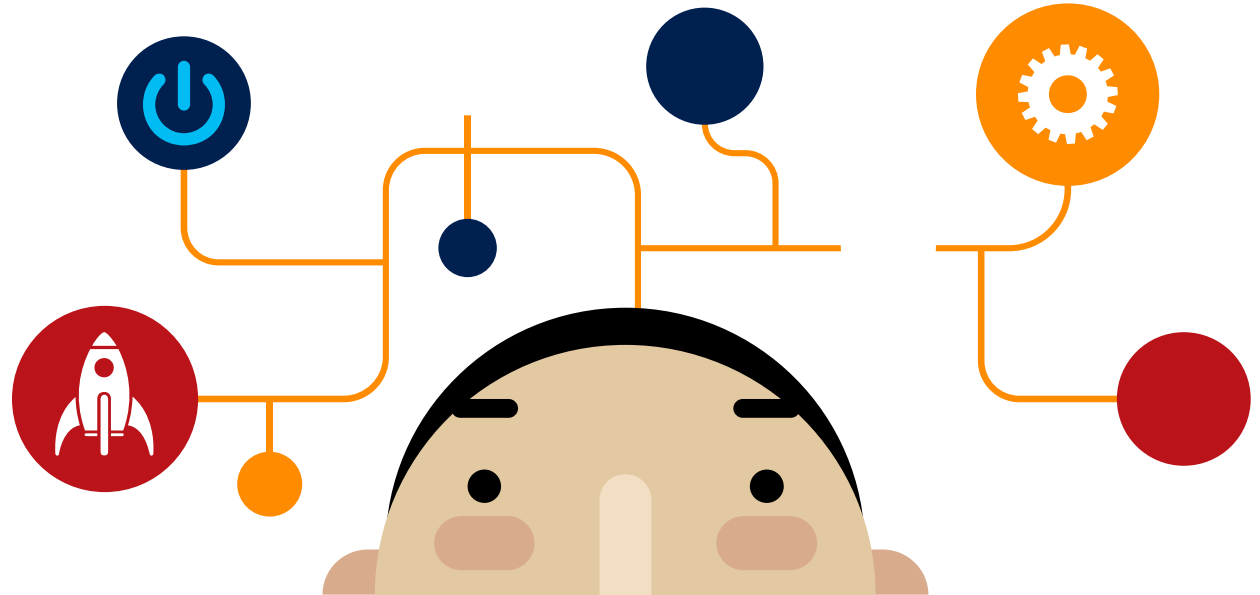
- ❑ O que é Python?
- ❑ Usando print
- ❑ Inserindo comentários
- ❑ Usando strings
- ❑ Trabalhando com números
- ❑ Trabalhando com datas
- ❑ Estruturas condicionais
- ❑ Utilizando coleções
- ❑ Laços de repetição
- ❑ Funções
- ❑ Módulos
- ❑ Pacotes

# O que é Python?



3

- Python é uma linguagem de programação de alto nível, interpretada e de propósito geral
- Foi criada por Guido van Rossum em 1991
- Enfatiza a clareza do código (*code readability*) que é feita por meio de indentação
- Suas construções de linguagem e sua abordagem orientada a objetos têm como objetivo ajudar os programadores a escrever código claro e bem estruturado



# O que é Python?



4



Linguagem de programação flexível



Projetada para ser “human readable”

# Por quê usar Python?



5



Linguagem amigável para iniciantes



Linguagem poderosa para usuários avançados



Comunidade atuante e interessada em ajudar

# O que é possível fazer com Python?



6



Modelos para aprendizado de máquina



Projetos de inteligência artificial



Aplicações Web



Ferramentas para automação



Praticamente qualquer aplicação computacional



print

# print



8

## Exibe informação no console

```
print('Hello world')
```

```
Hello world
```



# print



9

Strings podem ser representadas com aspas simples ou duplas

```
print('Hello world com aspas simples')
```

```
Print("Hello world com aspas duplas")
```

Hello world com aspas simples

Hello world com aspas duplas



# input / print

10

## Coletando informação do usuário

```
name = input('Digite seu nome: ')\nprint(name)
```

```
Digite seu nome: Pedro\nPedro
```

# print



11

Imprimir linhas em branco facilita a leitura dos dados

```
print('Hello world')
```

```
print()
```

```
print('Você notou a linha em branco?')
```

```
print('Quebra de linha \no meio de uma string')
```

Hello world

Você notou a linha em branco?

Quebra de linha

no meio de uma string

# print



12

## Debugando com print

```
print('Adicionando números')  
x = 42 + 206  
print('Realizando divisão')  
y = x / 0  
print('Operação concluída')
```

Adicionando números

Realizando divisão

Traceback (most recent call last):

File "demo.py", line 4, in <module>

y = x / 0

ZeroDivisionError: float division by zero



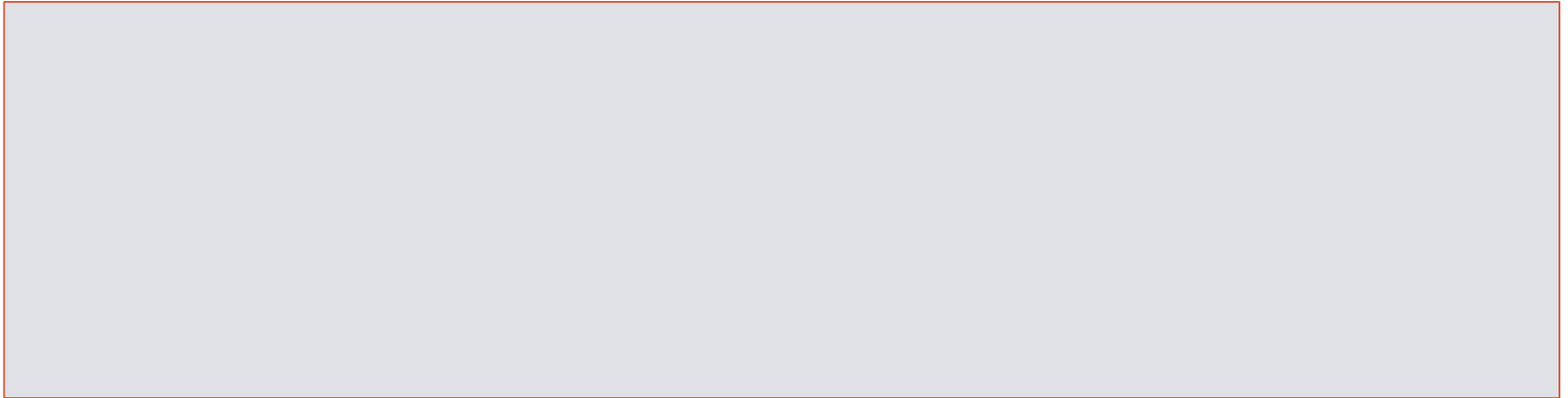
Comentários

# Comentários



14

# Esse é um comentário. Comentários são ignorados.



# Comentários



15

Comentários costumam ser usados para documentar o código de forma a facilitar seu entendimento

# O uso de aspas duplas se faz necessário quando a string

# contém aspas simples

```
print("It's a small world after all")
```

```
It's a small world after all
```



Strings



# Strings



17

Strings podem ser armazenadas em variáveis

```
nome = 'Pedro'
```

```
print(nome)
```

Pedro

# Strings



18

É possível concatenar strings com +

```
nome = 'Pedro'
```

```
sobrenome = 'Souza'
```

```
print(nome + sobrenome)
```

```
print('Olá ' + nome + ' ' + sobrenome)
```

```
PedroSouza
```

```
Olá Pedro Souza
```

# Strings



19

É possível utilizar funções para modificar strings

```
frase = 'O nome do aluno é Pedro'
```

```
print(frase.upper())
```

```
print(frase.lower())
```

```
print(frase.capitalize())
```

```
print(frase.count('o'))
```

O NOME DO ALUNO É PEDRO

o nome do aluno é pedro

O nome do aluno é pedro

4

# Strings



20

Funções auxiliam na formatação de strings

```
nome = input('Qual é seu nome? ')
```

```
sobrenome = input('Qual é seu sobrenome? ')
```

```
print('Olá ' + nome.capitalize() + ' ' + sobrenome.capitalize())
```

# Note o uso do input para fazer entrada de dados

```
Qual é seu nome? PEDRO
```

```
Qual é seu sobrenome? SOUZA
```

```
Olá Pedro Souza
```

# Strings



21

## Operador in (membership operator)

Permite verificar se uma substring existe numa string

```
txt = 'Eu moro em Araraquara'  
x = 'Arara' in txt  
print(x)
```

True

# Strings



22

Mais sobre strings

[https://www.w3schools.com/python/python\\_strings.asp](https://www.w3schools.com/python/python_strings.asp)

# Exercício 1



23

Leia duas strings e verifique se a segunda está contida na primeira.



Trabalhando com números





Assim como strings, números também podem ser armazenados em variáveis

Note que Python não exige a declaração de tipos para uma variável

```
pi = 3.14159  
print(pi)
```

```
3.14159
```

# Números



26

A combinação de strings com números não é permitida

```
dias_fev = 28
```

```
print(dias_fev + ' dias em Fevereiro')
```

File "demos.py", line 3, in <module>

```
print(dias_fev + ' dias em Fevereiro')
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'



Deve-se converter números em strings para permitir concatenação

```
dias_fev = 28
```

```
print(str(dias_fev) + ' dias em Fevereiro')
```

28 dias em Fevereiro



# Números / Strings

28

A função input sempre retorna strings

```
num1 = input('Digite o primeiro número: ')\nnum2 = input('Digite o segundo número: ')\nprint (num1 + num2)
```

```
Digite o primeiro número: 5\nDigite o segundo número: 6\n56
```



# Números / Strings

29

Números armazenados como strings precisam ser convertidos para números a fim de que seja possível realizar operações matemáticas

```
num1 = input('Digite o primeiro número: ')\nnum2 = input('Digite o segundo número: ')\nprint (int(num1) + int(num2))\nprint (float(num1) + float(num2))
```

Digite o primeiro número: 5

Digite o segundo número: 6

11

11.0



## Função para geração de um número randômico

```
import random # importante pacote  
print(random.randrange(1,10))
```

# Números



31

Mais sobre números

[https://www.w3schools.com/python/python\\_numbers.asp](https://www.w3schools.com/python/python_numbers.asp)

# Exercício 2



32

Leia dois números correspondentes a um intervalo, gere e imprima um número randômico dentro desse intervalo.





Datas



Python provê um módulo para manipulação de datas

```
# Temos que importar o módulo datetime
```

```
from datetime import datetime
```

```
data_corrente = datetime.now()
```

```
# a função now retorna um objeto datetime
```

```
print('Hoje é: ' + str(data_corrente))
```

```
Hoje é: 2020-02-07 16:17:18.694511
```



Existem funções que podem ser usadas com objetos `datetime` para manipulação de datas

```
from datetime import datetime, timedelta
data_corrente = datetime.now()
print('Hoje é: ' + str(data_corrente))
# timedelta é usado para definir um período de tempo
um_dia = timedelta(days=1)
ontem = data_corrente - um_dia
print('Ontem foi: ' + str(ontem))
```

```
Hoje é: 2020-02-07 16:17:18.694511
Ontem foi: 2020-02-06 16:17:18.694511
```



## Funções para formatação de datas

```
from datetime import datetime  
data_corrente = datetime.now()
```

```
print('Dia: ' + str(data_corrente.day))  
print('Mês: ' + str(data_corrente.month))  
print('Ano: ' + str(data_corrente.year))
```

```
Dia: 7  
Mês: 2  
Ano: 2020
```



# Datas (string -> datetime)

37

input devolve datas como strings – necessário converter

```
from datetime import datetime
```

```
nasc = input('Informe data nasc (dd/mm/yyyy): ')
```

```
data_nasc = datetime.strptime(nasc, '%d/%m/%Y')
```

```
print ('Nascimento: ' + str(data_nasc))
```

```
Informe data nasc (dd/mm/yyyy): 24/04/1998
```

```
Nascimento: 1998-04-24 00:00:00
```



## Outro exemplo de uso de funções

```
from datetime import datetime, timedelta
nasc = input('Qual é sua data de nascimento (dd/mm/yyyy)? ')
data_nasc = datetime.strptime(nasc, '%d/%m/%Y')
print ('Data do nascimento: ' + str(data_nasc))
um_dia = timedelta(days=1)
vespera_nasc = data_nasc - um_dia
print('Véspera do nascimento: ' + str(vespera_nasc))
```

```
Qual é sua data de nascimento (dd/mm/yyyy)? 24/04/1998
Data do nascimento: 1998-04-24 00:00:00
Véspera do nascimento: 1998-04-23 00:00:00
```

# Datas



39

Mais sobre datas

[https://www.w3schools.com/python/python\\_datetime.asp](https://www.w3schools.com/python/python_datetime.asp)

# Exercício 3



40

Leia uma data no passado e informe em qual dia da semana essa data caiu.





# Estruturas condicionais

# if



42

Código deve ser capaz de realizar diferentes ações de acordo com diferentes condições

```
if renda >= 1900:  
    imposto = 7.5  
    print(imposto)
```



# if / else

43

Para adicionar ações default use else

```
if renda >= 1900:
```

```
    imposto = 7.5
```

```
    print(imposto)
```

```
else:
```

```
    imposto = 0
```

```
    print imposto
```

# if / else



44

A indentação impacta a execução do código:

```
if renda >= 1900:
    imposto = 7.5
    print(imposto)
else:
    imposto = 0
    print(imposto)
```

```
if renda >= 1900:
    imposto = 7.5
else:
    imposto = 0
print(imposto)
```



# if – Múltiplas condições

45

Pode ser necessário checar múltiplas condições para tomar uma decisão. Considere uma empresa que cobra frete de acordo com o estado de entrega, conforme tabela a seguir

|         |       |
|---------|-------|
| SP e RJ | 10,00 |
| MG      | 15,00 |
| ES      | 18,00 |

Como escrever o código para calcular o custo do frete?



# if – Múltiplas condições

46

Como escrever o código para calcular o custo do frete?

```
if estado == 'SP':  
    frete = 10  
if estado == 'RJ':  
    frete = 10  
if estado == 'MG':  
    frete = 15  
if estado == 'ES':  
    frete = 18
```



# elif – Múltiplas condições

47

Se apenas uma condição pode ocorrer, é possível usar um único if em conjunto com um ou mais elif's

```
if estado == 'SP':  
    frete = 10  
elif estado == 'RJ':  
    frete = 10  
elif estado == 'MG':  
    frete = 15  
elif estado == 'ES':  
    frete = 18
```



# elif – Múltiplas condições

48

Usando elif é possível definir uma ação default

```
if estado == 'SP':  
    frete = 10  
elif estado == 'RJ':  
    frete = 10  
elif estado == 'MG':  
    frete = 15  
elif estado == 'ES':  
    frete = 18  
else:  
    frete = 20
```

|                |       |
|----------------|-------|
| SP e RJ        | 10,00 |
| MG             | 15,00 |
| ES             | 18,00 |
| Demais estados | 20,00 |





# elif – Múltiplas condições

49

Se há uma lista de valores a checar pode-se usar o operador **in**

```
if estado in('SP', 'RJ', 'PR'):
    frete = 10
elif estado == 'MG':
    frete = 15
elif estado == 'ES':
    frete = 18
else:
    frete = 20
```

|                       |              |
|-----------------------|--------------|
| SP, RJ e PR           | 10,00        |
| MG                    | 15,00        |
| ES                    | 18,00        |
| <b>Demais estados</b> | <b>20,00</b> |



# elif – Múltiplas condições

50

Considere agora que a empresa dá frete grátis para moradores de SP, RJ ou PR que tenham um cupom de desconto. Como expressar isso no código?

```
if estado in ('SP', 'RJ', 'PR') and cupom:  
    frete = 0
```

(Aqui estamos considerando que cupom é uma variável booleana)

# if / else



51

Mais sobre estruturas condicionais

[https://www.w3schools.com/python/python\\_conditions.asp](https://www.w3schools.com/python/python_conditions.asp)

# Exercício 4



52

Utilizando a tabela a seguir, escreva um código que permita obter a alíquota do imposto de renda de acordo com o valor da renda mensal. Seu programa deve ler o valor da renda e imprimir o valor da alíquota, bem como o valor do imposto a pagar.

| Renda                    | Alíquota |
|--------------------------|----------|
| De 1.903,99 até 2.826,65 | 7,5%     |
| De 2.826,66 até 3.751,05 | 15%      |
| De 3.751,06 até 4.664,68 | 22,5%    |
| Acima de 4.664,68        | 27,5%    |



Coleções



## Listas são coleções de itens

```
nomes = ['Pedro', 'Marina']  
notas = []  
notas.append(9.2) # Adiciona novo item no final  
notas.append(8.4)  
print(nomes)  
print(notas)  
print(notas[1]) # Coleções iniciam no zero
```

```
['Pedro', 'Marina']  
[9.2, 8.4]  
8.4
```



## Arrays também são coleções de itens

```
from array import array
notas = array('d')
notas.append(9.2)
notas.append(8.4)
print(notas)
print(notas[1])
```

```
array('d', [9.2, 8.4])
8.4
```



# Arrays vs Listas

56

Qual a diferença?



## Arrays

Armazenam tipos simples como números  
Todos os itens devem ser do mesmo tipo



## Lists

Armazenam qualquer dado  
De qualquer tipo





## Operações comuns

```
nomes = ['Pedro', 'Marina']  
print(len(nomes)) # Obtém o número de itens  
nomes.insert(0, 'Beto') # Insere antes do índice  
print(nomes)  
nomes.sort()  
print(nomes)
```

2

```
['Beto', 'Pedro', 'Marina']  
['Beto', 'Marina', 'Pedro']
```



## Manipulando intervalos

```
nomes = ['Pedro', 'Marina', 'Beto']  
alunos = nomes[0:2] # Obtém os dois primeiros itens  
# Índice inicial e número de itens a recuperar  
print(nomes)  
print(alunos)
```

```
['Pedro', 'Marina', 'Beto']  
['Pedro', 'Marina']
```



## Dicionários

```
peessoa = {'nome': 'Pedro'}  
peessoa['sobrenome'] = 'Souza'  
print(peessoa )  
print(peessoa ['nome'])
```

```
{'nome': 'Pedro', 'sobrenome': 'Souza'}  
Pedro
```

# Dicionários vs Listas



60

Qual a diferença?



***Dicionários***

Pares chave / valor

Ordem de armazenamento não garantido



***Listas***

Baseado em índice (inicia no 0)

Ordem de armazenamento garantida

# Coleções



61

## Mais sobre listas

[https://www.w3schools.com/python/python\\_lists.asp](https://www.w3schools.com/python/python_lists.asp)

## Mais sobre conjuntos

[https://www.w3schools.com/python/python\\_sets.asp](https://www.w3schools.com/python/python_sets.asp)

## Mais sobre dicionários

[https://www.w3schools.com/python/python\\_dictionaries.asp](https://www.w3schools.com/python/python_dictionaries.asp)



Laços



## Iterando numa coleção

```
for nome in ['Pedro', 'Marina', 'Beto']:  
    print(nome)
```

```
Pedro  
Marina  
Beto
```

# Laços



64

## Iterando um número de vezes

```
for index in range(0, 2):  
    print(index)
```

0

1





## Iteração controlada por uma condição

```
nomes = ['Pedro', 'Marina', 'Beto']
```

```
index = 0
```

```
while index < len(nomes):
```

```
    print(nomes[index])
```

```
    # Muda a condição
```

```
    index = index + 1
```

Pedro

Marina

Beto



## Iterando uma string

# Strings são listas de caracteres

```
for x in 'Ana':  
    print(x)
```

```
A  
n  
a
```

# Laços



67

Mais sobre laços while

[https://www.w3schools.com/python/python\\_while\\_loops.asp](https://www.w3schools.com/python/python_while_loops.asp)

Mais sobre laços for

[https://www.w3schools.com/python/python\\_for\\_loops.asp](https://www.w3schools.com/python/python_for_loops.asp)



- 5) Leia um conjunto de nomes e os armazene numa lista. Em seguida, leia um nome e verifique se o mesmo faz parte dessa lista.
- 6) Leia uma string e verifique se a mesma é um palíndromo.
- 7) Leia valores numéricos e os coloque numa lista. A leitura termina quando o valor 0 for digitado. Em seguida, calcule a média dos valores digitados e informe o usuário.



# Funções

# Funções



70

## Código com copy/paste

```
import datetime
# imprimir timestamps para ver o temp gasto
# para executar uma sequência de código
```

```
nome = 'Pedro'
print('Tarefa concluída')
print(datetime.datetime.now())
print()
```

```
for x in range(0,10):
    print(x)
print('Tarefa concluída')
print(datetime.datetime.now())
print()
```

```
Tarefa concluída
2020-02-13
16:55:01.815327
```

```
0
1
2
3
4
5
6
7
8
9
```

```
Tarefa concluída
20220-02-13
16:55:01.817263
```

# Funções



71

## Usar função para evitar repetição de código

```
import datetime
# Print the current time
def print_time():
    print('Tarefa concluída')
    print(datetime.datetime.now())
    print()
```

```
nome = 'Pedro'
print_time()
```

```
for x in range(0,10):
    print(x)
    print_time()
```

```
Tarefa concluída
2020-02-13 16:55:45.397319
```

```
0
1
2
3
4
5
6
7
8
9
```

```
Tarefa concluída
2020-02-13 16:55:45.399314
```



## Melhorando a função

# Importar a classe datetime da biblioteca datetime

```
from datetime import datetime
```

# Mostra a hora corrente

```
def print_time():
```

```
    print('Tarefa concluída')
```

# Agora não é necessário usar o prefixo datetime

```
    print(datetime.now())
```

```
    print()
```





## Se eu quiser imprimir diferentes mensagens

```
from datetime import datetime
nome = 'Pedro'
print('Nome atribuído')
print(datetime.now())
print()
```

```
for x in range(0,10):
    print(x)
print('Loop executado')
print(datetime.now())
print()
```

```
Nome atribuído
2020-02-14 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
Loop executado
2020-02-14 10:18:53.422748
```

# Funções



74

## Passando o nome da tarefa como parâmetro

```
from datetime import datetime
# Imprime timestamp e nome da tarefa
def print_time(nome_tarefa):
    print(nome_tarefa)
    print(datetime.now())
    print()

nome = 'Pedro'
print_time('Nome atribuído')

for x in range(0,10):
    print(x)
print_time('Loop executado')
```

```
Nome atribuído
2020-02-14 10:18:53.419754

0
1
2
3
4
5
6
7
8
9
Loop executado
2020-02-14 10:18:53.422748
```



## Funções com retorno

Vamos agora implementar uma função que receba uma string e retorne sua primeira letra. Utilizando essa função, vamos escrever um código que leia um nome e imprima as iniciais do nome lido.

# Funções



76

Função que retorna a primeira letra de uma string

```
def get_inicial(str):
```

```
    inicial = str[0:1]
```

```
    return inicial
```

```
nome = input('Digite seu nome: ')
```

```
inicial_nome = get_inicial(nome)
```

```
sobrenome = input('Digite seu sobrenome: ')
```

```
inicial_sobrenome = get_inicial(sobrenome)
```

```
print('Suas iniciais são: ' + inicial_nome + inicial_sobrenome)
```

```
Digite seu nome: Pedro  
Digite seu sobrenome: Souza  
Suas iniciais são: ps
```

# Funções



77

## Fazendo a função retornar inicial em maiúsculo

```
def get_inicial(str):
```

```
    inicial = str[0:1].upper()
```

```
    return inicial
```

```
nome = input('Digite seu nome: ')
```

```
inicial_nome = get_inicial(nome)
```

```
sobrenome = input('Digite seu sobrenome: ')
```

```
inicial_sobrenome = get_inicial(sobrenome)
```

```
print('Suas iniciais são: ' + inicial_nome + inicial_sobrenome)
```

```
Digite seu nome: Pedro
Digite seu sobrenome: Souza
Suas iniciais são: PS
```



## Considerações importantes:

- ▣ Funções tornam seu código mais legível e fácil de manter.
- ▣ Comentários devem ser usados para explicar o propósito de cada função.
- ▣ Funções devem ser declaradas antes da linha de código na qual são chamadas.

# Funções



79

## Funções podem ter múltiplos parâmetros

```
def get_inicial(str, maiuscula):  
    if maiuscula:  
        inicial = str[0:1].upper()  
    else:  
        inicial = str[0:1]  
    return inicial
```

```
nome = input('Digite seu nome: ')  
inicial_nome = get_inicial(nome, False)  
print('Sua inicial é: ' + inicial_nome)
```

```
Digite seu nome: Pedro  
Sua inicial é: p
```



É possível especificar um valor default para um parâmetro

```
def get_inicial(str, maiuscula=True):  
    if maiuscula:  
        inicial = str[0:1].upper()  
    else:  
        inicial = str[0:1]  
    return inicial
```

```
nome = input('Digite seu nome: ')  
inicial_nome = get_inicial(nome)  
print('Sua inicial é: ' + inicial_nome)
```

```
Digite seu nome: Pedro  
Sua inicial é: P
```





É possível especificar valores nomeando os parâmetros

```
def get_inicial(str, maiuscula):  
    if maiuscula:  
        inicial = str[0:1].upper()  
    else:  
        inicial = str[0:1]  
    return inicial
```

Quando os parâmetros são nomeado, eles podem aparecer em qualquer ordem



```
nome = input('Digite seu nome: ')  
inicial_nome = get_inicial(maiuscula=True, str=nome)  
print('Sua inicial é: ' + inicial_nome)
```

```
Digite seu nome: Pedro  
Sua inicial é: P
```

# Funções



82

Mais sobre funções

[https://www.w3schools.com/python/python\\_functions.asp](https://www.w3schools.com/python/python_functions.asp)

# Exercícios



83

- 8) Escreva uma função que receba um float representando o valor da temperatura em Celsius e retorne a temperatura equivalente em Farenheit. Em seguida, escreva um código que leia uma temperatura em Celsius e informe o valor equivalente em Farenheit.
- 9) Escreva uma função booleana que recebe uma string e verifica se a mesma é um palíndromo. Em seguida, escreva um código para ler uma string e, usando a função criada, verifique se a mesma é uma string.



## □ Exercício 10

Escreva uma **função** que receba como parâmetros valores de horas, minutos e segundos representando a duração de um evento e retorne o valor da duração totalizado em segundos.

Escreva uma **função** que transforme um valor de duração totalizado em segundos, para horas, minutos e segundos.

Escreva um **programa** que leia hora, minuto e segundo de entrada e de saída de um usuário na internet e, utilizando as funções de conversão, mostre na tela em horas, minutos e segundos o tempo que o usuário ficou conectado. Desconsidere a hipótese de que a desconexão tenha ocorrido após a meia-noite.