Programação em Python

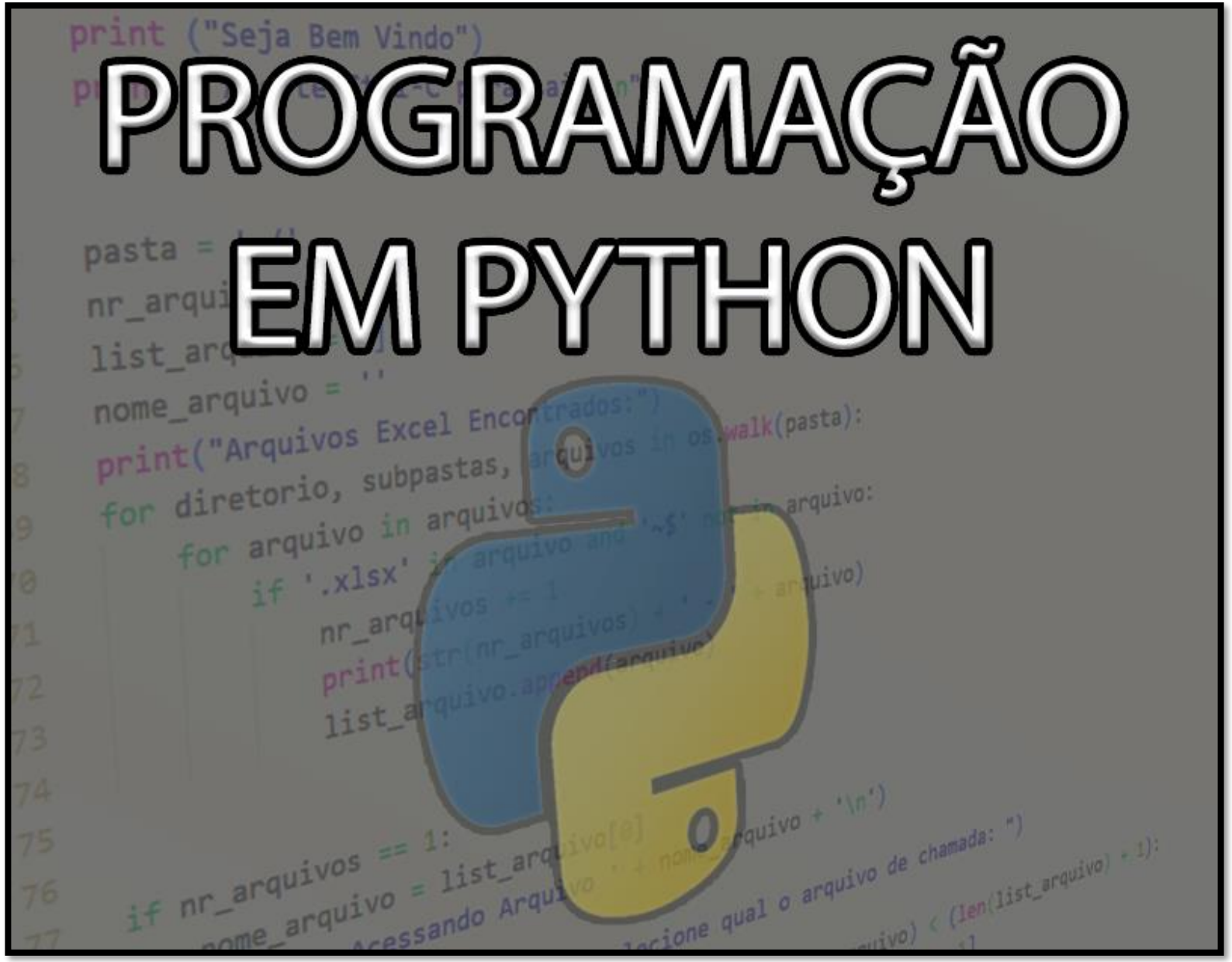
SENAI

Prof. Daniel Santos

daniel.sampaio@sp.senai.br

Senai Roberto Simonsen

R. Monsenhor Andrade, 298 - Brás, São Paulo - SP



Prof. Daniel Santos

Lidando com erros em Python

Quando criamos um programa, precisamos analisar se existe alguma maneira na qual o programa possa dar erro e encerrar inesperadamente.

Um exemplo no qual isso ocorre é ao chamar a função **print** e passar o valor de uma variável que não foi criada.

```
print(x)

print('Não vai executar, pois encerrou o programa inexperadamente')
```

```
Traceback (most recent call last):
  File "demo_try_except_error.py", line 3, in <module>
    print(x)
NameError: name 'x' is not defined
```

Neste caso ao executar o código **print(x)** recebemos um erro **NameError**

Lidando com erros em Python

Para lidar com os erros, podemos utilizar a instrução **try except**:

```
try:
  print(x)
except:
  print("Ocorreu um erro")

print('Apesar de ocorrer um erro, esse print será executado')
```

```
Ocorreu um erro
Apesar de ocorrer um erro, esse print será executado
```

Lidando com erros em Python

Também podemos realizar ações diferentes para diferentes tipos de erros, por exemplo:

```
x = 10
print('O número é' + x)
```

Esse código acima irá dar erro pois a variável **x** não é **string**

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
TypeError: can only concatenate str (not "int") to str
```

O tipo do erro é **TypeError**

Lidando com erros em Python

No código abaixo, lidamos especificamente com o erro do tipo **TypeError** e generalizamos quaisquer outros erros possíveis:

```
x = 10

try:
  print('O número é' + x)
except TypeError:
  print("Não pode concatenar int com string")
except:
  print("Tem outra coisa errada")
```

```
Não pode concatenar int com string
```

Lidando com erros em Python

No código abaixo, lidamos especificamente com o erro do tipo **TypeError** e generalizamos quaisquer outros erros possíveis:

```
x = 10

try:
  print('O número é' + x)
except TypeError:
  print("Não pode concatenar int com string")
except:
  print("Tem outra coisa errada")
```

```
Não pode concatenar int com string
```

Lidando com erros em Python

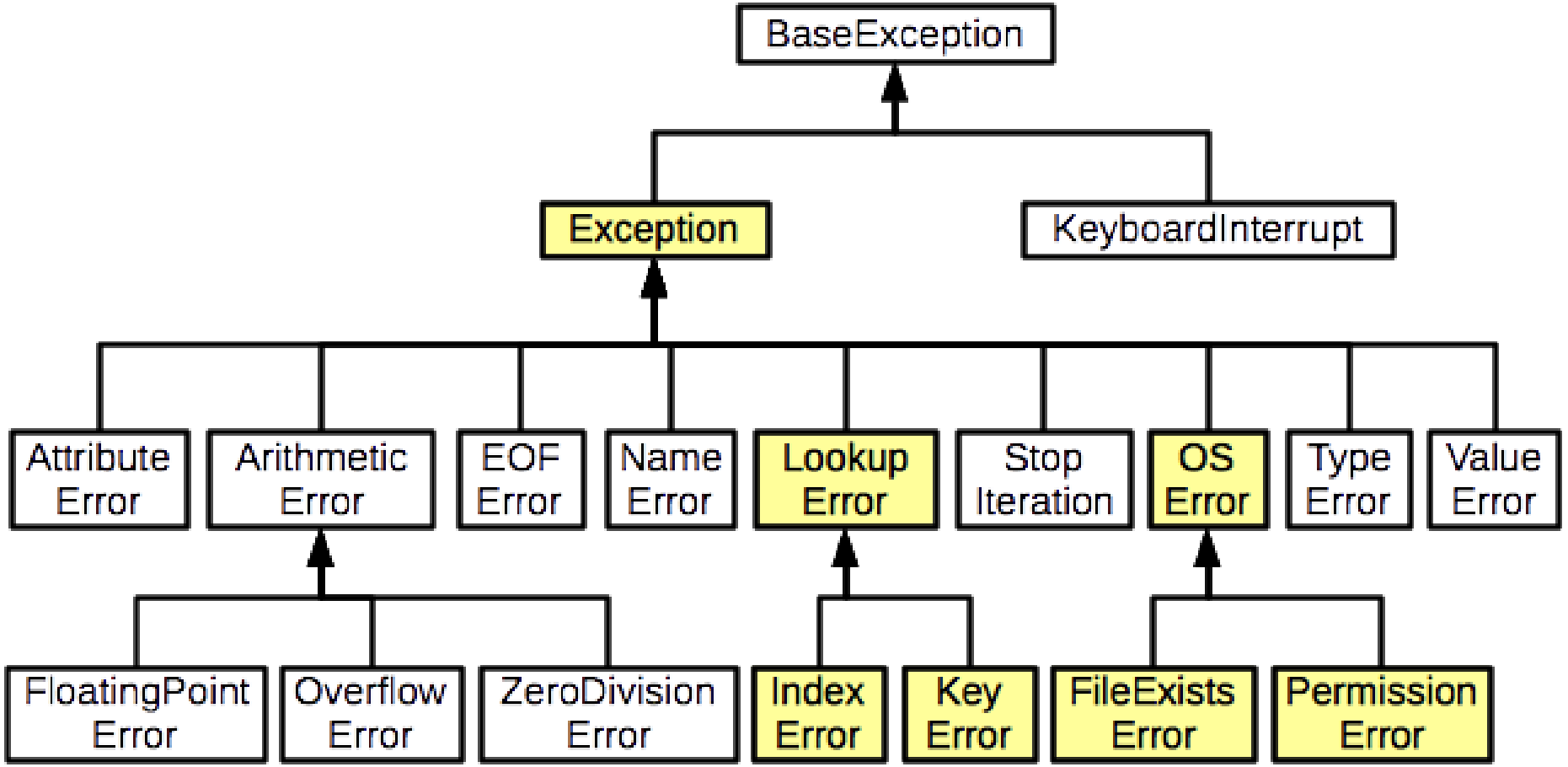
Também é possível criar erros não existentes internamente usando o comando **raise**:

```
idade = -5

if idade < 0:
  raise Exception("Não existe idade menor que zero")

print('Essa linha não será executada')
```

```
Traceback (most recent call last):
  File "./prog.py", line 5, in <module>
Exception: Não existe idade menor que zero
```

Mapa de Erros

Estes são os tipos de erros nativos em Python:



Funções

Podemos criar funções em python com ou sem parâmetros e com ou sem retorno.

Para criar uma função simples em python, basta usar a palavra-chave **def** como no exemplo

```
def minha_funcao():
  print("Comandos da função")
  print("Pode ter varias linhas de código")
  print("Dentro de uma mesma função")

minha_funcao()
```

Para chamar a função para executa-la, basta executar o nome dela com **()** no final.

Funções

Para criar uma função com parâmetros basta passar as informações desejadas dentro do parênteses:

```
def idade_em_2050(idade):
  print('Sua idade em 2050 será ' + str(2050-2021+idade) + " anos")

idade_em_2050(30)
idade_em_2050(15)
idade_em_2050(50)
```

```
Sua idade em 2050 será 59 anos
Sua idade em 2050 será 44 anos
Sua idade em 2050 será 79 anos
```

Neste exemplo, quando chamamos a função **idade_em_2050** passamos um argumento numérico (ex: **30**) e na criação da função esse valor numérico é atribuído ao parâmetro **idade**.

Funções

Podemos passar mais de um parâmetro dentro de uma função:

```
def idade_em_2050(nome, idade):
  print(nome + ', a sua idade em 2050 será ' + str(2050-2021+idade) + " anos")

idade_em_2050('João',30)
idade_em_2050('Maria',15)
idade_em_2050('José',50)
```

```
João, a sua idade em 2050 será 59 anos
Maria, a sua idade em 2050 será 44 anos
José, a sua idade em 2050 será 79 anos
```

Funções

Também podemos colocar palavras-chave nos parâmetros, assim não precisamos nos preocupar com a ordem dos argumentos passados:

```
def idade_em_2050(idade, nome):
  print(nome + ', a sua idade em 2050 será ' + str(2050-2021+idade) + " anos")

idade_em_2050(nome = 'João', idade = 30)
idade_em_2050(idade = 15, nome = 'Maria')
idade_em_2050(nome = 'José', idade = 50)
```

```
João, a sua idade em 2050 será 59 anos
Maria, a sua idade em 2050 será 44 anos
José, a sua idade em 2050 será 79 anos
```

Funções

É possível definir parâmetros com valores padrões (**default**), pra quando não for passado aquele argumento ter um valor pré-definido:
OBS: Nestes casos, na hora de criar a função, coloque os parâmetros que terão valores **default** por último

```
def idade_em_2050(idade, nome = 'Zé Ninguém'):
  print(nome + ', a sua idade em 2050 será ' + str(2050-2021+idade) + " anos")

idade_em_2050(nome = 'João', idade = 30)
idade_em_2050(idade = 15, nome = 'Maria')
idade_em_2050(nome = 'José', idade = 50)
idade_em_2050(idade = 20)
```

```
João, a sua idade em 2050 será 59 anos
Maria, a sua idade em 2050 será 44 anos
José, a sua idade em 2050 será 79 anos
Zé Ninguém, a sua idade em 2050 será 49 anos
```

Funções

Podemos definir retorno para funções e salvar o valor retornado ou utiliza-lo diretamente em outra função:

```
def idade_em_2050(idade):
  return 2050-2021 + idade

print(idade_em_2050(30))

idade_jose = idade_em_2050(25)
print(idade_jose)
```

```
59
54
```

```
def nome_ao_contrario(nome):
  nome_invertido = ''
  for indice, letra in enumerate(nome):
    nome_invertido += nome[-indice-1]
  return nome_invertido

print(nome_ao_contrario('Daniel'))
print(nome_ao_contrario('João'))
print(nome_ao_contrario('Josinaldo'))
```

```
leinaD
oãoJ
odlanisoJ
```

OBS: A função **enumerate** utilizada no exemplo serve para conseguir retornar o índice em um loop For, neste caso, como o loop For é de uma string, o índice será a posição de cada letra da palavra.

Funções

Podemos chamar uma função dentro da própria função:

```
def teste(letra, quant):
  if(quant > 0):
    print(letra*quant)
    teste(letra,quant - 1)

teste('a',22)
```

```
aaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaa
aaaaaaaaaaaaaa
aaaaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaa
aaaaaaaaaa
aaaaaaaaa
aaaaaaaa
aaaaaaa
aaaaaa
aaaaa
aaaa
aaa
aa
a
```