

Paradigma funcional com python

DOCENTE: PROF. LUÍS EDUARDO TENÓRIO SILVA
DISCENTE: NAYANE JACYARA CANDIDO SANTOS



Introdução

PYTHON É UMA LINGUAGEM DE PROGRAMAÇÃO QUE SUPORTA O PARADIGMA FUNCIONAL, JUNTAMENTE COM O PARADIGMA IMPERATIVO E O PARADIGMA ORIENTADO A OBJETOS. EMBORA PYTHON NÃO SEJA ESTRITAMENTE UMA LINGUAGEM FUNCIONAL, ELA FORNECE RECURSOS QUE PERMITEM ESCREVER CÓDIGO FUNCIONAL.

ALGUMAS DAS CARACTERÍSTICAS DA PROGRAMAÇÃO FUNCIONAL EM PYTHON INCLUEM:

FUNÇÕES PURAS: FUNÇÕES QUE NÃO TÊM EFEITOS COLATERAIS E SEMPRE PRODUZEM O MESMO RESULTADO PARA O MESMO CONJUNTO DE ENTRADAS. ESSAS FUNÇÕES NÃO ALTERAM O ESTADO DO PROGRAMA E NÃO DEPENDEM DO ESTADO DO PROGRAMA.

FUNÇÕES DE ORDEM SUPERIOR: FUNÇÕES QUE RECEBEM OUTRAS FUNÇÕES COMO ARGUMENTOS OU RETORNAM FUNÇÕES COMO RESULTADO. ISSO PERMITE QUE AS FUNÇÕES SEJAM COMPOSTAS DE MANEIRA MODULAR.

EXPRESSÕES LAMBDA: FUNÇÕES ANÔNIMAS QUE PODEM SER CRIADAS EM TEMPO DE EXECUÇÃO. ESSAS FUNÇÕES SÃO GERALMENTE USADAS COMO ARGUMENTOS PARA OUTRAS FUNÇÕES DE ORDEM SUPERIOR.

ITERADORES: OBJETOS QUE PERMITEM ITERAR SOBRE UMA SEQUÊNCIA DE ELEMENTOS DE MANEIRA PREGUIÇOSA, O QUE SIGNIFICA QUE OS ELEMENTOS SÃO GERADOS SOB DEMANDA À MEDIDA QUE SÃO NECESSÁRIOS.

COMPREENSÃO DE LISTA: UMA MANEIRA CONCISA DE CRIAR LISTAS A PARTIR DE OUTRAS LISTAS OU SEQUÊNCIAS, USANDO UMA SINTAXE QUE É SEMELHANTE À NOTAÇÃO MATEMÁTICA PARA CONJUNTOS.

EM RESUMO, A PROGRAMAÇÃO FUNCIONAL EM PYTHON É SUPORTADA POR MEIO DE RECURSOS COMO FUNÇÕES PURAS, FUNÇÕES DE ORDEM SUPERIOR, EXPRESSÕES LAMBDA, ITERADORES E COMPREENSÃO DE LISTA. ESSES RECURSOS PERMITEM ESCREVER CÓDIGO MAIS LEGÍVEL, CONCISO E MODULAR.

Funções de alta ordem:

SÃO FUNÇÕES QUE RECEBEM OUTRAS FUNÇÕES COMO ARGUMENTOS OU RETORNAM FUNÇÕES COMO RESULTADO. ISSO PERMITE QUE AS FUNÇÕES SEJAM COMPOSTAS DE MANEIRA MODULAR.

```
1 #Funções de alta ordem
2 from functools import reduce
3
4 def soma(x, y):
5     return x + y
6
7 numeros = [6, 75, 9, 30]
8 soma_total = reduce(soma, numeros)
9 print("soma_total:", soma_total)
```

Composição de funções:

A COMPOSIÇÃO DE FUNÇÕES PODE SER USADA PARA SIMPLIFICAR O CÓDIGO E AUMENTAR A LEGIBILIDADE, ALÉM DE PERMITIR A REUTILIZAÇÃO DE FUNÇÕES EM DIFERENTES CONTEXTOS.

```
1  #Composição de funções
2
3  alunos = [{'nome': 'Julia', 'nota': 10},
4            {'nome': 'Julyane', 'nota': 7.0},
5            {'nome': 'Lucas', 'nota': 6.0},
6            {'nome': 'Ana', 'nota': 9.0},
7            {'nome': 'Joao', 'nota': 5.5}]
8
9  aprovados = list(filter(lambda x: x['nota'] >= 7, alunos))
10
11 print("Aprovados:", aprovados)
```

TERMINAL SAÍDA CONSOLE DE DEPURAÇÃO PROBLEMAS

```
PS C:\Users\Jacyara> & C:/Users/Jacyara/AppData/Local/Programs/Python/Python39/python.exe c:/Users/Jacyara/wo
ao/AtividadeDeApresentacao.py
Aprovados: [{'nome': 'Julia', 'nota': 10}, {'nome': 'Julyane', 'nota': 7.0}, {'nome': 'Ana', 'nota': 9.0}]
```

Imutabilidade:

SÃO FUNÇÕES QUE RECEBEM OUTRAS FUNÇÕES COMO ARGUMENTOS OU RETORNAM FUNÇÕES COMO RESULTADO. ISSO PERMITE QUE AS FUNÇÕES SEJAM COMPOSTAS DE MANEIRA MODULAR.

```
1  # Imutabilidade
2
3  numeros = [5, 12, 16, 20]
4
5  new_numeros = list(map(lambda x: x * 2, numeros))
6
7  print("numeros:", numeros)
8  print("new_numeros:", new_numeros)
```

TERMINAL

SAÍDA

CONSOLE DE DEPURAÇÃO

PROBLEMAS

```
PS C:\Users\Jacyara> & C:/Users/Jacyara/AppData/Local/Programs/Python/Python39-64/AtividadeDeApresentacao.py
numeros: [5, 12, 16, 20]
new_numeros: [10, 24, 32, 40]
```

Funções puras

SÃO FUNÇÕES QUE NÃO TÊM EFEITOS COLATERAIS E SEMPRE PRODUZEM O MESMO RESULTADO PARA O MESMO CONJUNTO DE ENTRADAS. ESSAS FUNÇÕES NÃO ALTERAM O ESTADO DO PROGRAMA E NÃO DEPENDEM DO ESTADO DO PROGRAMA.

```
1  #Funções puras
2
3  lista = [-10, 32, -44, 64, -25]
4
5  numeros_positivos = list(filter(lambda x: x > 0, lista))
6  print("numeros_positivos:", numeros_positivos)
7
```

TERMINAL SAÍDA CONSOLE DE DEPURAÇÃO PROBLEMAS

PS C:\Users\Jacyara> & C:/Users/Jacyara/AppData/Local/Programs/Python/Python39-64/AtividadeDeApresentacao.py
numeros_positivos: [32, 64]

Recursão:

RECURSÃO É UM CONCEITO EM PROGRAMAÇÃO ONDE UMA FUNÇÃO SE CHAMA A SI MESMA, REPETIDAMENTE, ATÉ QUE UMA CONDIÇÃO DE PARADA SEJA ATINGIDA. ISSO SIGNIFICA QUE A FUNÇÃO USA SEU PRÓPRIO RESULTADO COMO ENTRADA PARA A PRÓXIMA CHAMADA, CRIANDO UMA SEQUÊNCIA DE CHAMADAS DE FUNÇÃO QUE SE AUTO-REFERENCIAM.

```
1  # Recursão
2
3  def factorial(n):
4      if n == 1:
5          return 1
6      else:
7          return n * factorial(n-1)
8  print("factorial:",factorial(3))
```

TERMINAL

SAÍDA

CONSOLE DE DEPURAÇÃO

PROBLEMAS

```
PS C:\Users\Jacyara> & C:/Users/Jacyara/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:/Users/Jacyara/AppData/Local/Programs/Python/Python39-64/Scripts/AtividadeDeApresentacao.py
factorial: 6
```


Currying:

É UMA FUNÇÃO QUE PODE SER APLICADA PARCIALMENTE, RETORNANDO UMA NOVA FUNÇÃO QUE ESPERA O ARGUMENTO RESTANTE.

```

1 # Curryng
2
3 def saudacao(falar):
4     def MeuNome(name):
5         return f"{falar} {name}!"
6     return MeuNome
7
8 resultado = saudacao("Oi, meu nome é")("Nayane")
9 print(resultado)
10
11 falarIngles = saudacao("Hi, I am")
12 falarPortugues = saudacao("Olá, eu sou")
13
14 print(falarIngles("José"))
15 print(falarPortugues("Maria"))
16
```

TERMINAL SAÍDA CONSOLE DE DEPURAÇÃO PROBLEMAS

PS C:\Users\Jacyara> & C:/Users/Jacyara/AppData/Local/Programs/Python/P...
ao/AtividadeDeApresentacao.py
Oi, meu nome é Nayane!
Hi, I am José!
Olá, eu sou Maria!