
targettrust
treinamento e tecnologia

Funções e Lambda λ

Recapitulando ...

```python
def valida_login(usuario, senha):
  if(usuario == 'matheus' and senha == '12345'):
    return True
  else:
    return False

user  = input('Digite o usuário: ')
password = input('Digite a senha: ')

if valida_login(user, password):
  print('Acesso permitido!')
else:
  print('Acesso negado!')
```

```python
def mostra_menu():
  print('-----------------------------------')
  print('1 - Cadastro de clientes')
  print('2 - Cadastro de vendas')
  print('3 - Sair')
  print('-----------------------------------')

opcao = 0

while opcao != 3:
  mostra_menu()
  opcao = int(input('Digite uma opção: '))

print('Saindo...')
```

Lambda λ

```
def soma(x, y):
    return x + y

subtracao = lambda x, y: x - y

print(soma(5, 3))
print(subtracao(5, 3))
```

Prática

Dado um conjunto de dados, como eliminar os outliers? Ou seja, os pontos fora da curva.

a) Calcular a média

b) Calcular o desvio padrão

c) Criar uma nova lista apenas com os dados dentro do intervalo [media – desvio, media + desvio]

15 min

Prática

Transforme os exercícios a seguir em funções:

1. Escreva um algoritmo em Python que lê 3 valores e retorna o maior deles.

2. Escreva um algoritmo em que recebe o salário, hora trabalhadas e qtd dias e retorne o salário líquido.

15 min

Filter / Map / Reduce / Zip

- *A função filter(função, lista) oferece uma maneira conveniente de filtrar todos os elementos de um iterável para o qual a função retorne True.*
- *A função filter(função( ), l) precisa de uma função como seu primeiro argumento. A função precisa retornar um valor booleano (True ou False).*
- *Esta função será aplicada a cada elemento do iterable. Somente se a função retornar True, o elemento do iterable será incluído no resultado.*

Filter / Map / Reduce / Zip

- *map () é uma função que leva em dois argumentos: uma função e uma seqüência iterable. Na forma:*

  map(função, sequência)

- *O primeiro argumento é o nome de uma função e a segunda uma seqüência (por exemplo, uma lista).*

- *map() aplica a função a todos os elementos da seqüência. Ele retorna uma nova lista com os elementos alterados por função.*

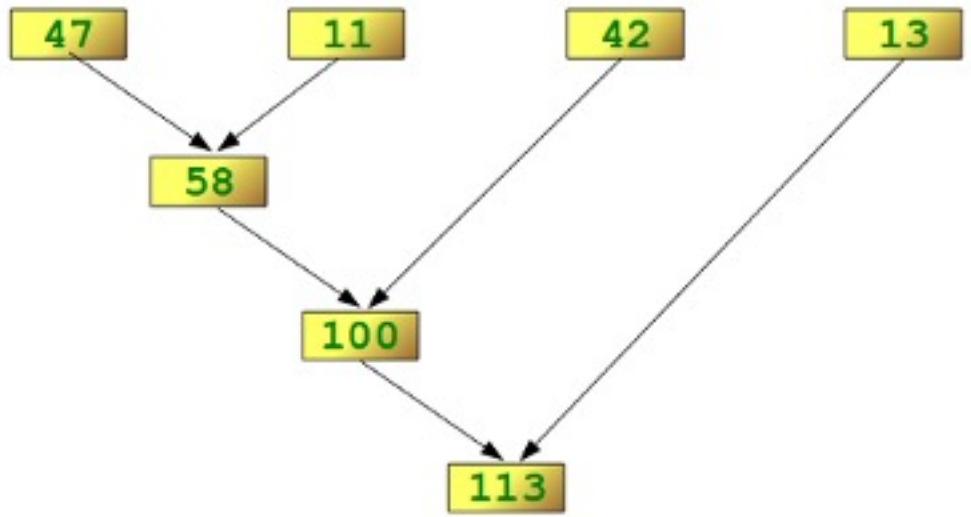
- *map() pode ser aplicado a mais de um iterable. Os iteráveis devem ter o mesmo comprimento.*

- *Por exemplo, se estamos trabalhando com duas listas-map() aplicará sua função lambda aos elementos das listas de argumentos, ou seja, aplica-se primeiro aos elementos com o índice 0, e depois aos elementos com o 1° índice até o que o índice N seja alcançado.*

Filter / Map / Reduce / Zip

- *A função reduce(função, sequência) aplica continuamente a função à sequência. Em seguida, ele retorna um único valor.*
- *Se seq = [s1, s2, s3, ..., sn], a redução de chamada (função, seqüência) funciona assim:*
  - *No início, os dois primeiros elementos de seq serão aplicados à função, isto é, func(s1, s2)*
  - *A lista em que a reduce() funciona parece assim: [função (s1, s2), s3, ..., sn]*
  - *No próximo passo, a função será aplicada no resultado anterior e no terceiro elemento da lista, ou seja, função(função (s1, s2), s3)*
  - *A lista parece agora: [função (função (s1, s2), s3), ..., sn]*
  - *Continua assim até apenas um elemento é deixado e retorna esse elemento como resultado de reduzir ()*

Filter / Map / Reduce / Zip



Filter / Map / Reduce / Zip

- *zip() cria um iterador que agrega elementos de cada um dos iteráveis.*
- *Retorna um iterador de tuplas, onde a i-ésima tupla contém o i-ésimo elemento de cada uma das seqüências ou do iterável passado como argumento.*
- *O iterador para quando a entrada mais curta disponível se esgota. Com um único argumento iterável, ele retorna um iterador de n-tuplas. Sem argumentos, ele retorna um iterador vazio.*
- *O zip() só deve ser usado com entradas de comprimento desiguais quando você não se preocupa com os valores ininterruptos, além dos valores mais longos.*

Unindo com Lambda λ

```
notas_EDI = [
  [1.0, 7.1, 8.5, 7.5, 0.0, 4.0, 9.5, 4.5, 8.0, 7.0, 9.0, 3.0, 9.5, 10.0, 6.0, 9.0, 6.0,
7.0],
  [6.0, 9.5, 7.0, 9.5, 7.5, 8.5, 7.0, 0.0, 5.0, 9.0, 10.0, 7.0, 8.5, 0.0, 8.5, 7.5, 5.5, 1
0.0]
]

medias = list(map(lambda x: round(sum(x) / len(x), 1), notas_EDI))
print(medias)
```

Unindo com Lambda λ

```
alunos = ["ALYSSON", "ANDRINO", "BRUNA", "EDGAR", "GABRI
EL", "HENRIQUE", "JEAN", "JOÃO", "KAUAN", "LUAN", "LUCAS", "
LUIZ", "MATEUS", "RICARDO", "RODRIGO", "VINICIUS", "WAGNE
R", "WILLIAM"]

alunos_com_nome_pequeno = list(filter(lambda x: len(x) <= 5, a
lunos))
print(alunos_com_nome_pequeno)
```

targettrust
treinamento e tecnologia

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br


targettrust
treinamento e tecnologia

Enumerate

- Enumerate permite você contar elementos enquanto itera através de um objeto. O método retorna uma tupla na forma (contagem, elemento).

- enumerate( ) torna-se particularmente útil quando você precisa ter algum tipo de rastreador. Por exemplo:

```
for count,item in enumerate(lst):
    if count >= 2:
        break
    else:
        print item
```

List / Dict Comprehension

- Além das operações de sequência e métodos de lista, o Python inclui uma operação mais avançada chamada de compreensão de lista ou dicionário.

- As compreensões de lista ou dicionário nos permitem construir esses objetos usando uma notação diferente. Você pode pensar nisso essencialmente como um loop construído dentro de colchetes.

```
# Pega todas as letras em uma string
lst = [x for x in 'word']
```

Prática

Crie uma lista de 3 elementos e calcule a terceira potência de cada elemento usando List Comprehension.

05 min

Prática

Crie duas funções, uma para elevar um número ao quadrado e outra para elevar ao cubo. Aplique as duas funções aos elementos da lista abaixo.

Obs: as duas funções devem ser aplicadas simultaneamente.

05 min

Prática

Usando a função filter( ), encontre os valores que são comuns às duas listas abaixo.

10 min

Prática

Considere a lista abaixo e retorne apenas os elementos cujo índice for maior que 5.

```
lista = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

10 min

Prática

Considere os dois dicionários abaixo.
Crie um terceiro dicionário com as chaves do dicionário 1 e os valores do dicionário 2.

```
dict1 = {'a':1,'b':2}
dict2 = {'c':4,'d':5}
```

10 min

Prática

Reescreva o código abaixo, usando a função map(). O resultado final deve ser o mesmo!

```
palavras = 'Minha terra tem palmeiras onde canta o Sabiá, As aves, que aqui gorjeiam, Não gorjeiam como lá.'.split()

resultado = [[w.upper(), w.lower(), len(w)] for w in palavras]

for i in resultado:
    print (i)
```

15 min

Tratamento de Exceções

```
print('Hello)
```

Python

```
  File "<ipython-input-1-db8c9988558c>", line 1
    print('Hello)
                 ^
SyntaxError: EOL while scanning string literal
```

- Observe como obtemos um SyntaxError, com a descrição adicional de que era um EOL (End of Line Error). Isso é suficiente para que percebamos que esquecemos um único apóstrofe no final da linha. Compreender estes vários tipos de erro irá ajudá-lo a depurar seu código muito mais rápido.

Tratamento de Exceções

- Este tipo de erro e descrição é conhecido como uma Exceção. Mesmo que uma declaração ou expressão seja sintaticamente correta, pode causar um erro quando tentamos executá-la. Os erros detectados durante a execução são chamados de exceções e não são incondicionalmente fatais.

- Você pode verificar a lista completa de exceções embutidas (https://docs.python.org/2/library/exceptions.html). Vamos aprender a lidar com erros e exceções em nosso próprio código.

Tratamento de Exceções

- A terminologia básica e a sintaxe usadas para lidar com erros no Python são as instruções **try** e **except**. O código que pode causar uma exceção ocorre é colocado no bloco *try* e o tratamento da exceção é implementado no * do bloco de código *except*. Sintaxe é:

```
try:
        Você tenta fazer algo aqui...
        ...
except ExceptionI:
        Se causar a ExceptionI, roda isso.

except ExceptionII:
        Se causar a ExceptionII, roda isso.
...
else:
     Se não causar excessões, roda isso.
```

Tratamento de Exceções

- Agora, e se continuássemos querendo executar código após a ocorrência da exceção? É aí que o **finally** entra.

- Usando o finally: o bloco de código sempre será executado, independentemente de existir uma exceção no bloco de código try. A sintaxe é:

```
try:
    Seu código aqui
    ...
    Devido a qualquer exceção, este código pode ser ignorado!

finally:
    Este bloco de código sempre seria executado.
```

Datetime

- O Python possui o módulo de datetime para ajudar a lidar com timestamps em seu código. Os valores de tempo são representados com a classe de time. Time têm atributos por hora, minuto, segundo e microssegundo. Eles também podem incluir informações de fuso horário. Os argumentos para inicializar uma instância de time são opcionais, mas é improvável que o padrão de 0 seja o que você deseja.

- O tempo de data também nos permite trabalhar com timestamps de data. Os valores da data do calendário são representados com a classe de data. As instâncias possuem atributos por ano, mês e dia.

- É fácil criar uma data que represente a data de hoje usando o método de classe today().

Lendo e Escrevendo em Arquivos

Expressões Regulares

- Expressões regulares são padrões de correspondência de texto descritos com uma sintaxe formal.
- Muitas vezes você ouvirá expressões regulares referidas como 'regex' ou 'regexp' na conversa.
- As expressões regulares podem incluir uma variedade de regras, a busca de repetição, a correspondência de texto e muito mais. Ao avançar no Python, você verá que muitos dos seus problemas de análise podem ser resolvidos com expressões regulares.

targettrust
treinamento e tecnologia

Obrigado!

Erro no material? Envie e-mail para:
materiais@targettrust.com.br