Curso: Machine Learning Básico com Python

Aula 01 - PYTHON para Inteligência Artificial

Roteiro 02 – Funções, entradas e formatação de dados e controle de fluxo de execução

Este roteiro comporta a construção de funções, leitura e formatação de dados e estruturas de controle de fluxo de execução.

1. Funções

O uso de funções permite incluir na análise de dados estruturas de condição e iteração. A seguir é apresentado o primeiro exemplo de uma função que tem sua definição a partir da instrução "def" com o nome da função logo a seguir e os parâmetros entre parênteses.

```
def soma(a,b):
  c=a+b
  return c

soma(5,4)
```

2. Entrada de dados e formatação

Um dos recursos que as vezes precisamos considerar é o de receber valores de quem está usando a aplicação. Isto permite que as funções e conjunto de instruções em Python no Colab se tornem mais genéricas sem necessidade de alterá-las a todo o momento para alterar determinados valores. Para isso é possível utilizar a instrução "input". A instrução "input" para a execução da célula ou função, solicita ao usuário que digite um determinado valor e após a digitação do valor atribui para uma eventual variável o valor digitado.

Veja o exemplo:

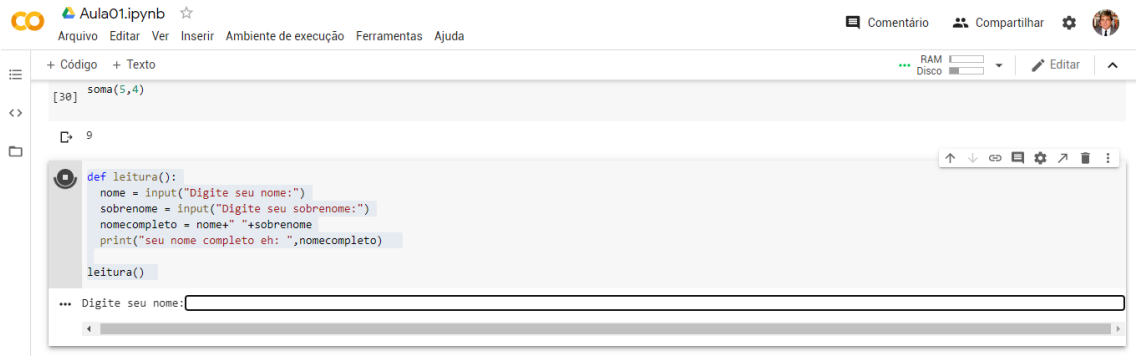
```
def leitura():
  nome = input("Digite seu nome:")
  sobrenome = input("Digite seu sobrenome:")
  nomecompleto = nome+" "+sobrenome
  print("seu nome completo eh: ",nomecompleto)

leitura()
```

```
Digite seu nome:joao
Digite seu sobrenome:Souza
seu nome completo eh:  joao Souza
```

Note que quando você executa o código a execução para a cada instrução input e apresenta uma caixa de texto a ser preenchida conforme a figura 1:

Figura 1. Caixa de texto de entrada de dados



Esta será a forma padrão de eventuais entradas de dados. Note que a variável nome e sobrenome se tornam variáveis tipo String e por isso se a entrada de dados necessita ser de um tipo específico é necessário fazer a conversão indicando o tipo para o qual a entrada deve ser transformada. Para isso, utiliza-se o int (string) para converter para o tipo inteiro, ou float (string) para converter para o tipo float. Veja os exemplos:

Exemplo com tipo de dados int (inteiro):

```
def potencia(a,b):
  c=a**b
  return c

x = int(input("Digite um valor:"))
y = int(input("Digite o valor de potência:"))
print(potencia(x,y))
```

```
Digite um valor:3
Digite o valor de potência:4
81
```

Outro exemplo agora com o tipo de dados float (ponto flutuante):

```
def soma(a,b):
  c=a+b
  return c

x = float(input("Digite um valor:"))
y = float(input("Digite um valor:"))
print(soma(x,y))
```

```
Digite um valor:3.24
Digite um valor:5.4
8.64
```

Em termos de formatação existem duas formas de contornar os problemas com números em ponto flutuante em termos de apresentação. Uma é usando um recurso do Python 2 que é o formato entre aspas no print. Outra forma de resolver é usando a função format, ou ainda usar a função round. Como nosso foco não é este não vamos discutir o problema em detalhes, mas seguem alguns exemplos de formatação.

a) format para String: “:[preencher][alinhar][largura].[precisão]”.format(<string>)

Onde:

[preencher]: Qualquer caractere.

[alinhar]: “<” para alinhar à esquerda, “>” à direita e “^” ao centro.

[largura]: Largura mínima do campo.

[precisão] Largura máxima do campo.

Veja um exemplo:

```
s = 'Teste String'
# alinha a direita com 20 espaços em branco
print("{0:>20}".format(s))
# alinha a direita com 20 símbolos #
print("{0:#>20}".format(s))
# alinha ao centro com 10 " " a esquerda e 10 " " a direita
print("{0:^20}".format(s))
# imprime só as primeiras cinco letras
print("{0:.5}".format(s))
```

```
        Teste String
########Teste String
    Teste String
Teste
```

b) format para números: “:[preencher][alinhar][sinal][largura].[precisão][tipo]”.format(<num>)

Onde:

[preencher]: Qualquer caractere.

[alinhar]: “<” para alinhar à esquerda, “>” à direita e “^” ao centro.

[sinal]: + se apresentar

[largura]: Largura mínima do campo.

[precisão] Largura máxima do campo.

[tipo]: d ou i (inteiro), f ou F (float), o (octal), x ou X (hexadecimal), e ou E (exponencial)

Veja o exemplo:

```
print("{0:4}".format(-123))
# aparece '-123'
print("{0:4}".format(123))
# aparece ' 123'
print("{0:4.2f}".format(33.3287))
# aparece '33.33'
print("{0:+4.2f}".format(33.3287))
# aparece '+33.33'
print("{0:+4.2e}".format(33.3287))
```

```
# aparece '+3.33e+01'
print("{0:b}".format(123))
# aparece '1111011'
```

```
-123
 123
33.33
+33.33
+3.33e+01
1111011
```

Quem estiver interessado em ler mais sobre o problema de ponto flutuante leia sobre isso em:

https://docs.python.org/pt-br/3/tutorial/floatingpoint.html

Para o controle do fluxo de execução em funções no Python existem instruções específicas que indicam as condições e iterações. A seguir conheceremos estas estruturas.

3. Estruturas de Decisão (Condicionais)

As estruturas de decisão permitem alterar o fluxo de execução de um programa, percorrendo um ou outro conjunto de instruções de acordo com o valor (Verdadeiro/Falso) de um teste lógico. Em Python temos as estruturas de decisão “se” (if), “se/senão” (if..else) e “se/senão se/senão” (if..elif..else)

A instrução if é utilizado quando precisamos decidir se um trecho do programa deve ou não ser executado. Ele é associado a uma condição, e o trecho de código será executado se o valor da condição for verdadeiro.

Sintaxe:

if <condição>:
    <instruções>

Exemplo:

```
nota = float(input("Digite sua nota na disciplina:"))
if nota < 6.0:
  print('Reprovado!')
```

```
Digite sua nota na disciplina:5.5
Reprovado!
```

Na instrução if..else um trecho de código será executado se a condição for verdadeira e outro se a condição for falsa.

Sintaxe:

if <condição1>:
    <instruções1>
else :
    <instruções2>

Exemplo:

```
nota = float(input("Digite sua nota na disciplina:"))
if nota >= 6.0:
  print('Aprovado!')
else:
  print('Reprovado!')
```

```
Digite sua nota na disciplina:5.66
Reprovado!
```

A instrução if..elif..else é usada quando houver diversas condições, cada uma associada a um trecho de código.

Sintaxe:

if <condição1>:
    <instruções1>
elif <condição2>:
    <instruções2>
elif <condição3>:
    <instruções3>
...
else :
    <instruçõesN>

Note que somente o bloco de comandos associado à 1a condição verdadeira encontrada será executado. Se nenhuma das condições tiver valor verdadeiro, executa o bloco de comandos default.

Exemplo:

```
idade = int(input("Digite sua idade:"))
if idade < 3:
  print('Bebê')
elif idade < 10:
  print('Infantil')
elif idade < 14:
  print('Junior')
elif idade < 18:
  print('Adolescente')
elif idade < 30:
  print('Jovem')
else:
  print('Adulto')
```

```
Digite sua idade:10
Junior
```

4. Estruturas de Repetição

A Estrutura de repetição também conhecida como “loop” é utilizada para executar uma sequência de comandos por várias vezes. A repetição está associada ou a uma condição, que

indica se deve continuar a repetição, ou a uma sequência de valores, que determina quantas vezes a sequência deve ser repetida.

4.1 Laço while

No laço while, o trecho de código da repetição está associado a uma condição. Enquanto a condição tiver valor verdadeiro, o trecho é executado. Quando a condição passa a ter valor falso, a repetição termina.

Sintaxe:

while <condição>:
  <instruções>

Exemplo:

```
senha = "54321"
leitura =" "
while (leitura != senha):
    leitura = input("Digite a senha: ")
    if leitura == senha : print('Acesso liberado ')
    else: print('Senha incorreta. Tente novamente')
```

```
Digite a senha: 12345
Senha incorreta. Tente novamente
Digite a senha: abcde
Senha incorreta. Tente novamente
Digite a senha: 54321
Acesso liberado
```

Exemplo: Encontrar a soma de 5 valores.

```
contador = 0
somador = 0
while contador < 5:
     contador = contador + 1
     valor = float(input('Digite o '+str(contador)+'° valor: '))
     somador = somador + valor
print('Soma = ', somador)
```

```
Digite o 1° valor: 1
Digite o 2° valor: 2
Digite o 3° valor: 3
Digite o 4° valor: 4
Digite o 5° valor: 5
Soma =  15.0
```

4.2 Laço for

O laço for é a estrutura de repetição mais utilizada em Python. Pode ser utilizado com uma sequência numérica (gerada com o comando range) ou associado a uma lista. O trecho de código da repetição é executado para cada valor da sequência numérica ou da lista.

Sintaxe:

for <variável> in range (<início>, <limite>, <passo>):
  <instruções>
ou

for <variável> in <lista>:
  <instruções>

Exemplos:

a) Encontrar a soma S = 1+4+7+10+13+16+19

```
S=0
for x in range(1,20,3):
     S = S+x
print('Soma = ',S)
```

```
Soma =  70
```

b) As notas de um aluno estão armazenadas em uma lista. Calcular a média dessas notas.

```
Lista_notas= [3.4,6.6,8,9,10,9.5,8.8,4.3]
soma=0
for nota in Lista_notas:
     soma = soma+nota
média = soma/len(Lista_notas)
print('Média = ', '{:.4f}'.format(média))
```

```
Média =  7.4500
```

É importante notar que todas estas estruturas de controle de fluxo de execução podem estar dentro das funções que aprendemos a declarar. Veja exemplos anteriores implementados como funções que permitem a execução com qualquer coleção de dados de entrada.

```
def CalcMedia(lista):
  soma=0
  for nota in lista:
    soma = soma+nota
  return soma/len(lista)

Lista_notas= [3.4,6.6,8,9,10,9.5,8.8,4.3]
media = CalcMedia(Lista_notas)
print('Média = ', '{:.4f}'.format(media))
```

```
Média =  7.4500
```

No caso de estruturas de decisão isso também é verdade. Veja como fica como função o código que indica a categoria a partir da idade.

```
def categoria(idade):

  if idade < 3:
    ctg = 'Bebê'
  elif idade < 10:
    ctg = 'Infantil'
  elif idade < 14:
    ctg = 'Junior'
  elif idade < 18:
    ctg = 'Adolescente'
  elif idade < 30:
    ctg = 'Jovem'
  else:
    ctg = 'Adulto'
  return ctg

idd = int(input("Digite sua idade:"))
print("Sua categoria é:", categoria(idd))
```

```
Digite sua idade:23
Sua categoria é: Jovem
```

Já uma função “MaiorValor” para retornar o maior número de uma lista precisa de usar tanto as estruturas de repetição quanto as estruturas de repetição. Veja:

```
def MaiorValor(lista):
  if len(lista) > 0:
    maior = lista[0]
  else:
    maior = 0;
  for valor in lista:
    if valor > maior:
      maior = valor
  return maior

ListaNotas= [3.4,6.6,8,9,10,9.5,8.8,4.3]
resultado = MaiorValor(ListaNotas)
print(resultado)
```

```
10
```

Para ler uma lista basta usar a instrução “for” com o “range”. Veja esta função “Leitura” que recebe uma lista e inclui uma determinada quantidade de valores inteiros na lista:

```
def Leitura(lista,qtd):
  for i in range(1,qtd+1):
    lista.append(int(input("Digite o valor do elemento ")))

ListaNotas= []
Leitura(ListaNotas,5)
```

```
print(ListaNotas)
```

```
Digite o valor do elemento 1
Digite o valor do elemento 2
Digite o valor do elemento 3
Digite o valor do elemento 4
Digite o valor do elemento 5
[1, 2, 3, 4, 5]
```

Agora veremos um exemplo de uso de funções com dicionários. Nestes exemplos criamos uma agenda simples com nomes e telefones:

```
def cria():
  d = {}
  return d

def inclui(d):
  nome = input("Digite o nome a incluir: ")
  if nome in d:
    print("Nome já existe não pode ser incluído")
  else:
    fone = input("Digite o fone: ")
    d[nome]=fone

def consulta(d):
  nome = input("Digite o nome a consultar: ")
  print("Telefone: ",d.get(nome, 'Não disponível'))

def altera(d):
  nome = input("Digite o nome do fone a alterar: ")
  print("Telefone atual: ",d.get(nome, 'Não disponível'))
  if nome in d:
    fone = input("Digite o novo fone: ")
    d[nome]=fone
    print("Telefone alterado!")
  else:
    print("Telefone não pode ser alterado!")

def exclui(d):
  nome = input("Digite o nome a excluir: ")
  if nome in d:
    del d[nome]
    print("Nome e fone excluídos!")
  else:
    print("Nome Inexistente!")

def mostra(d):
  print("Agenda:")
  for nome in d.keys():
    print(nome,d[nome])
```

```
agenda = cria()
inclui(agenda)
inclui(agenda)
inclui(agenda)
consulta(agenda)
altera(agenda)
exclui(agenda)
mostra(agenda)
```

```
Digite o nome a incluir: Marcelo
Digite o fone: 11111
Digite o nome a incluir: Helena
Digite o fone: 22222
Digite o nome a incluir: Marcelo
Nome já existe não pode ser incluído
Digite o nome a consultar: Cynthia
Telefone:  Não disponível
Digite o nome do fone a alterar: Helena
Telefone atual:  22222
Digite o novo fone: 33333
Telefone alterado!
Digite o nome a excluir: Cynthia
Nome Inexistente!
Agenda:
Marcelo 11111
Helena 33333
```

Uma possível alternativa para o programa principal é a apresentação de um menu de opções:

```
agenda = cria()
opc = 10
while opc != 0:
  opc = int(input("Digite 0.Sair,1.Inc,2.Con,3.Alt,4.Exc,5.Most:"))
  if opc == 1:
    inclui(agenda)
  elif opc == 2:
    consulta(agenda)
  elif opc == 3:
    altera(agenda)
  elif opc == 4:
    exclui(agenda)
  elif opc == 5:
    mostra(agenda)
  elif opc == 0:
    print("Bye!")
  else:
    print("Opção inválida!")
```