

Blindando Entradas em Python

Mário Leite

...

Como já havia mencionado aqui, em postagens anteriores, o MAIS importante na criação e desenvolvimento de qualquer programa de computador são as entradas dos dados; seja um simples calculo de média escolar até sofisticados sistemas de reconhecimento facial (o *Big Brother* no futuro da humanidade). Se qualquer uma dessas entradas não estiver correta (validadas) as chances de o resultado sair errado são muito grandes. Por isto o programador iniciante deve estar bem atendo a esta situação e esquecer, por um minuto, que programar é saber a diferença entre *front-end* e *back-end*".

Em Python existem várias maneiras de validar entradas nos programas: seja através de funções como *type()* ou de métodos como *isnumeric()* e *isdigit()*. Entretanto, é muito importante saber EXATAMENTE como funcionam seus parâmetros e seus retornos; caso contrário pode dar muito errado quando isto não for observado. Na verdade, segundo a “Lei de Murphy”, em algum momento vai ocorrer um erro usando estas palavras-chave se não forem rigidamente observadas, mesmo que você confie nos exemplos na Internet bem arrumadinhos!

Então, o que deve ser feito para “blindar” erros de entradas no Python!?

Existe um recurso nas linguagens mais modernas de programação que resolve o problema definitivamente, mesmo com as opções mostradas acima: é a “Estrutura de Tratamento de Erros”; a famosa “**try..except**”. Observe a versão **V1** do programa “**RaizesEquGrau2** onde foi usado o método *isnumeric()* para verificar o tipo de dado dos coeficientes (**a**, **b**, **c**) de uma equação do segundo grau: quando um dos coeficientes é um valor numérico negativo o retorno desse método é **False**, o que impede o fluxo do programa a sair do *loop*. estranhamente, pois esta validação está errada (qualquer um dos coeficientes pode ter qualquer valor, exigindo apenas que o coeficiente **a** seja diferente de zero. Na **figura 1a** quando TODOS os coeficientes são positivos e a saída é normal; sem problema. Entretanto, como mostra a **figura 1b**, se for digitado um valor negativo para qualquer um desses coeficientes (o que é válido matematicamente) ocorre uma realimentação do *loop* sem sentido algum! Isto ocorre porque o método *isnumeric()* retorna **False** se o valor for negativo!

Na versão **V2** as entradas foram devidamente “blindadas” com o emprego de uma estrutura “**try..except**” de maneira rígida na validação dos dados e dispensando o emprego do método “*isnumeric()*”, como também as conversões dos coeficientes para *float* após o *loop*. Observe na **figura 2a** uma saída normal quando todos os coeficientes com valores positivos, abandonando o *loop* para fazer os devidos cálculos e mesmo com valor negativo para o coeficiente **b**. Na **figura 2b** o *loop* de entrada continua sendo executado ao detectar um valor não numérico para o coeficiente **b**, mesmo não empregando *isnumeric()*.

Então, mesmo que você veja exemplos bem “arrumadinhos” na Internet sobre como verificar tipos de dados com funções e métodos, não se esqueça de que na prática “o buraco é mais embaixo” e SEMPRE que puder deve usar estrutura de tratamento de erros para blindar as entradas de dados lidos pelo teclado; pode ser feia, mas funciona!

```

'''
RaizesEquGrau2_V1.py
Em Python 3.9
Calcula as raízes de uma equação do segundo grau.
-----
Autor: Mário Leite
Data: 19/03/2023
-----
'''

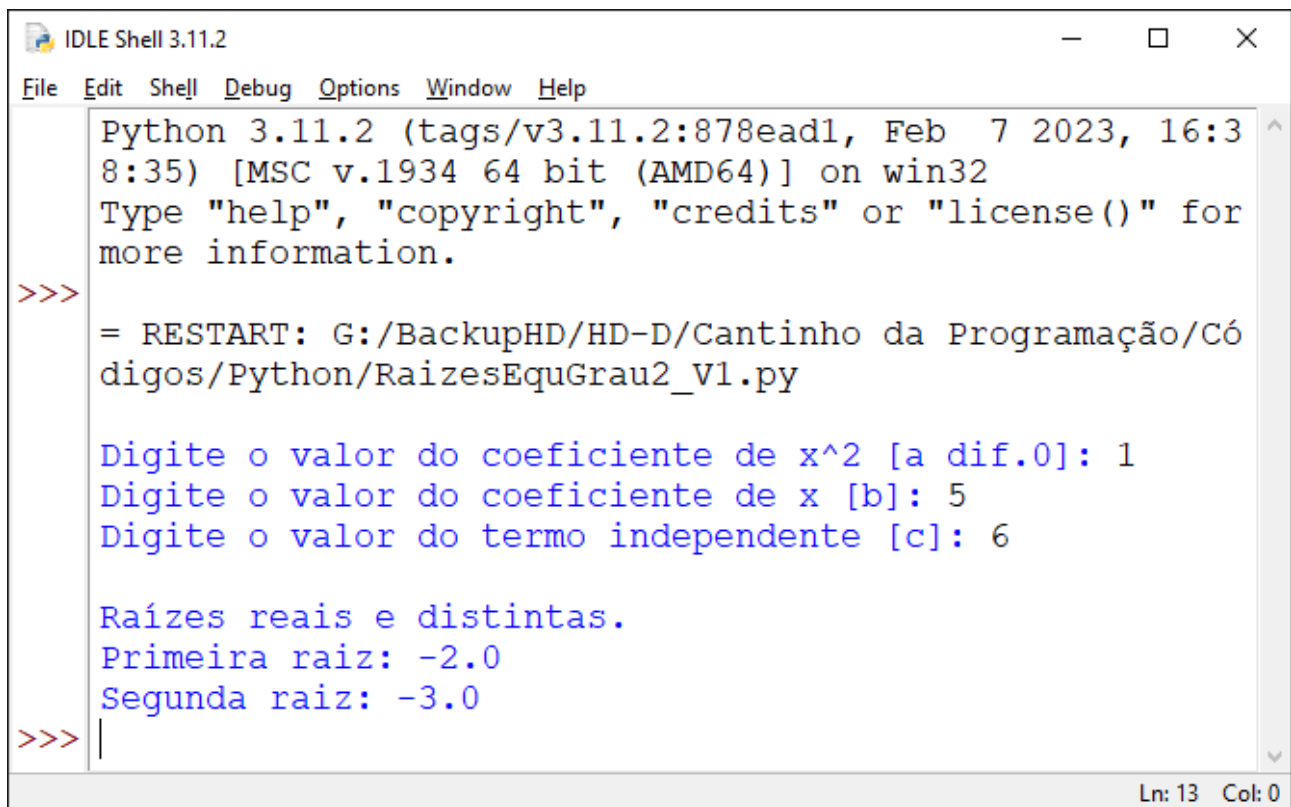
import math

cond = True
while (cond):
    a = input("Digite o valor do coeficiente de x^2 [a dif.0]: ")
    b = input("Digite o valor do coeficiente de x [b]: ")
    c = input("Digite o valor do termo independente [c]: ")
    cond1 = not(a.isnumeric())
    cond2 = not(b.isnumeric())
    cond3 = not(c.isnumeric())
    cond = ((a=="0") or (cond1) or (cond2) or (cond3))
    print()
#Fim da validação das entradas

a = float(a)
b = float(b)
c = float(c)
delta = b**2 - (4*a*c)

print()
if(delta > 0):
    print("Raízes reais e distintas.")
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}')
    print(f'Segunda raiz: {x2}')
elif(delta == 0):
    print("Raízes reais e iguais.")
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}')
    print(f'Segunda raiz: {x2}')
else:
    print("Raízes complexas.")
    delta = abs(delta)
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}{ "i" }')
    print(f'Segunda raiz: {x2}{ "i" }')
#Fim do programa "RaizesEquGrau2_V1"-----

```



The screenshot shows the IDLE Shell 3.11.2 window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell displays the Python 3.11.2 startup message. The user has restarted the shell and run a script. The script prompts for coefficients of a quadratic equation. The user enters 1, 5, and 6. The script outputs the roots: -2.0 and -3.0.

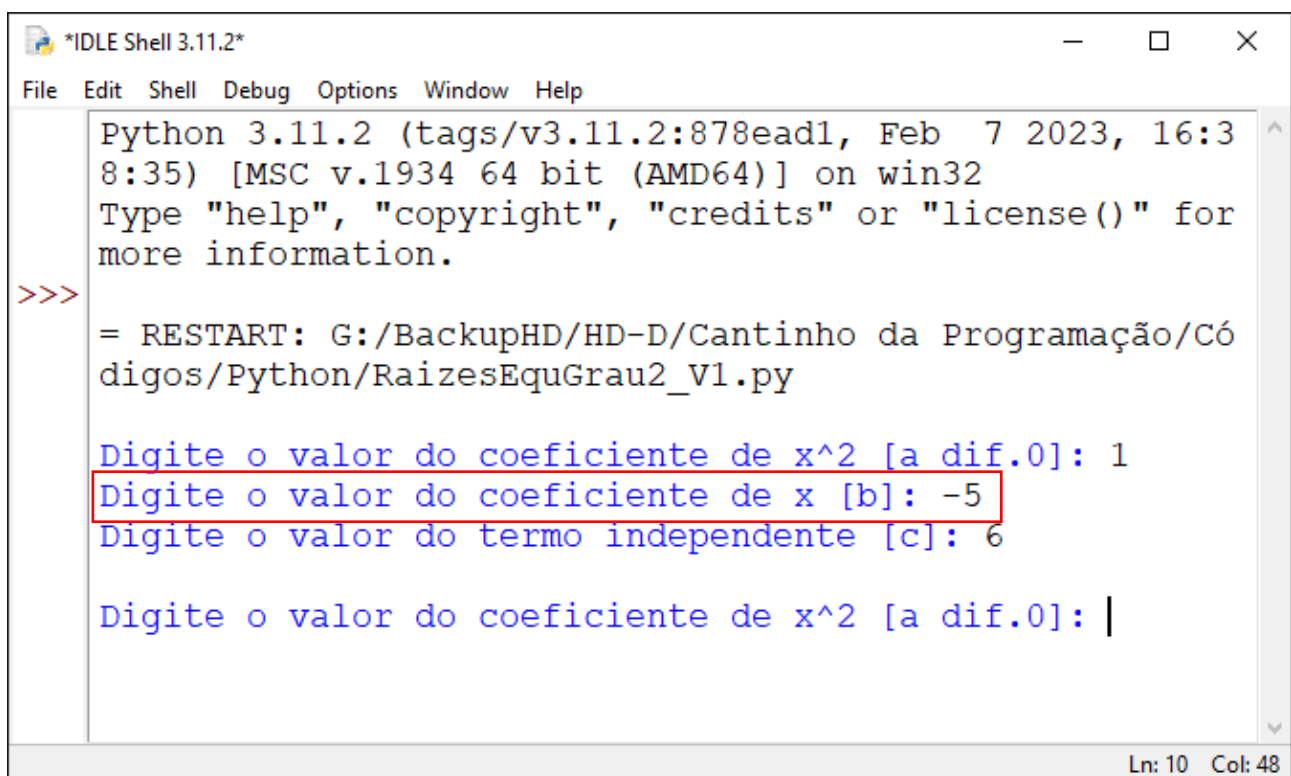
```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:/BackupHD/HD-D/Cantinho da Programação/Códigos/Python/RaizesEquGrau2_V1.py

Digite o valor do coeficiente de x^2 [a dif.0]: 1
Digite o valor do coeficiente de x [b]: 5
Digite o valor do termo independente [c]: 6

Raízes reais e distintas.
Primeira raiz: -2.0
Segunda raiz: -3.0
>>> |
```

Ln: 13 Col: 0

Figura 1a - Saída normal com a Versão 1



The screenshot shows the IDLE Shell 3.11.2 window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell displays the Python 3.11.2 startup message. The user has restarted the shell and run a script. The script prompts for coefficients of a quadratic equation. The user enters 1, -5, and 6. The script outputs the roots: -2.0 and -3.0. The input -5 is highlighted with a red box.

```
*IDLE Shell 3.11.2*
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: G:/BackupHD/HD-D/Cantinho da Programação/Códigos/Python/RaizesEquGrau2_V1.py

Digite o valor do coeficiente de x^2 [a dif.0]: 1
Digite o valor do coeficiente de x [b]: -5
Digite o valor do termo independente [c]: 6

Digite o valor do coeficiente de x^2 [a dif.0]: |
```

Ln: 10 Col: 48

Figura 1b - Saída anormal: impedindo uma entrada válida

```

'''
RaizesEquGrau2_V2.py
Em Python 3.9
Calcula as raízes de uma equação do segundo grau.
-----

Autor: Mário Leite
Data: 19/03/2023
-----

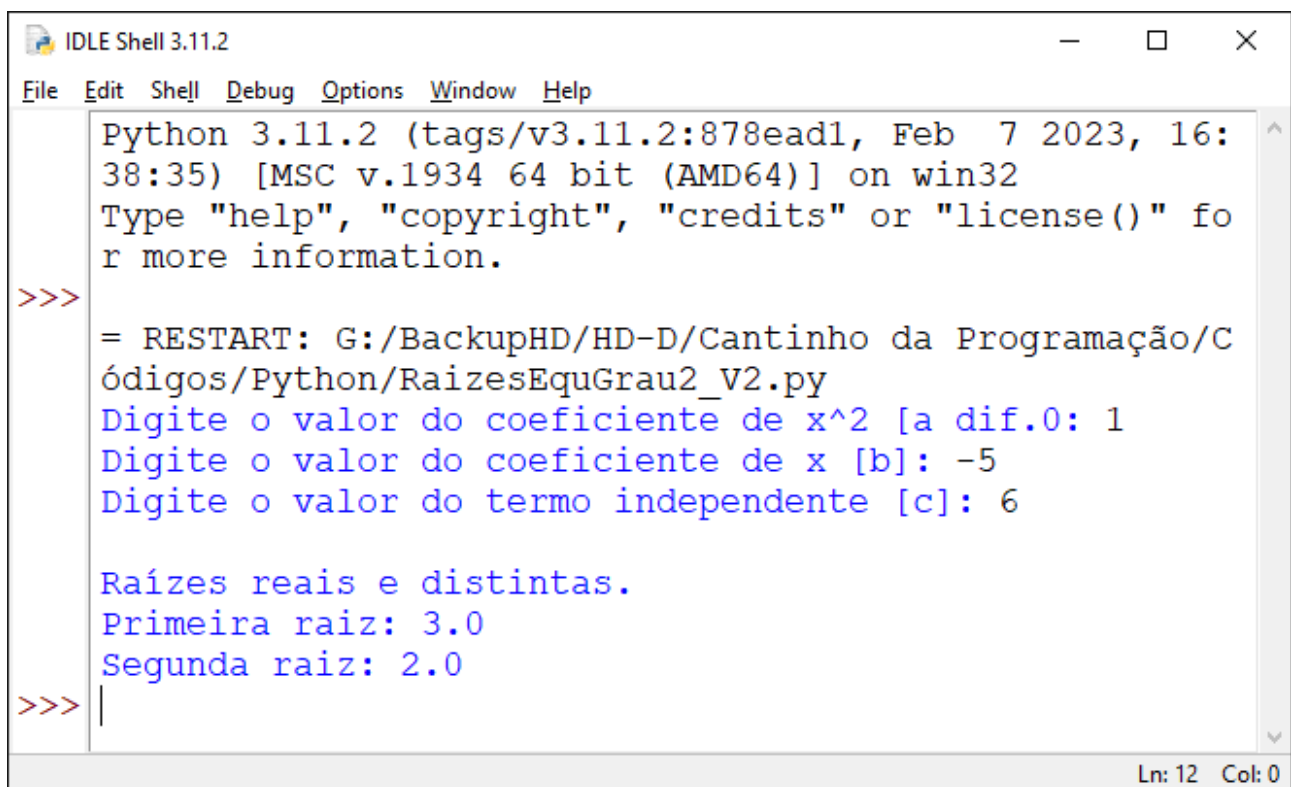
'''

import math

print()
cond = True
while (cond):
    try:
        a = float(input("Digite o valor do coeficiente de x^2 [a dif.0:]"))
        b = float(input("Digite o valor do coeficiente de x [b]: "))
        c = float(input("Digite o valor do termo independente [c]: "))
        if (a != 0):
            break
        print()
    except:
        print()
        cond = True #força a realimentação do loop
#Fim da validação das entradas

delta = b**2 - (4*a*c)
print()
if (delta > 0):
    print("Raízes reais e distintas.")
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}')
    print(f'Segunda raiz: {x2}')
elif (delta == 0):
    print("Raízes reais e iguais.")
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}')
    print(f'Segunda raiz: {x2}')
else:
    print("Raízes complexas.")
    delta = abs(delta)
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    print(f'Primeira raiz: {x1}{ "i" }')
    print(f'Segunda raiz: {x2}{ "i" }')
#Fim do programa "RaizesEquGrau2_V2"-----

```



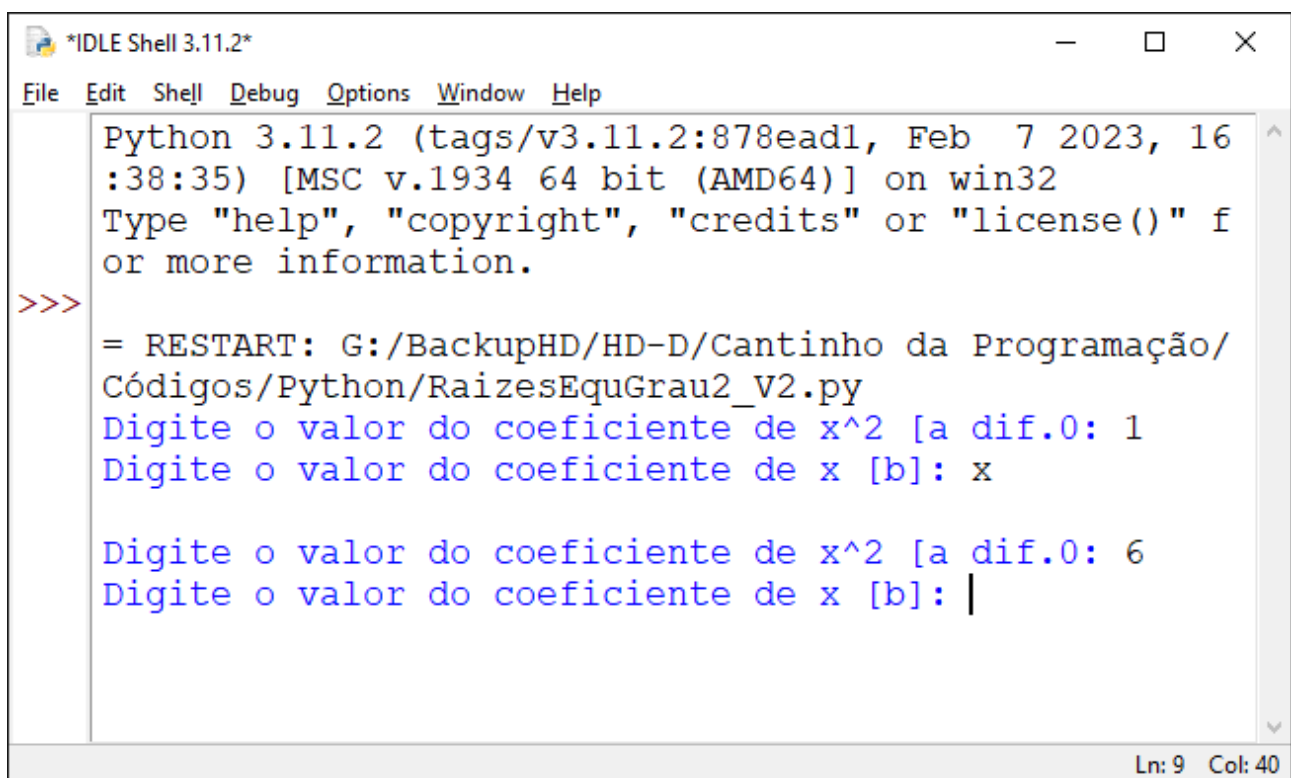
```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: G:/BackupHD/HD-D/Cantinho da Programação/Códigos/Python/RaizesEquGrau2_V2.py
Digite o valor do coeficiente de x^2 [a dif.0: 1
Digite o valor do coeficiente de x [b]: -5
Digite o valor do termo independente [c]: 6

Raízes reais e distintas.
Primeira raiz: 3.0
Segunda raiz: 2.0
>>> |
```

Ln: 12 Col: 0

Figura 2a - Saída normal: mesmo com coeficiente negativo



```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> = RESTART: G:/BackupHD/HD-D/Cantinho da Programação/Códigos/Python/RaizesEquGrau2_V2.py
Digite o valor do coeficiente de x^2 [a dif.0: 1
Digite o valor do coeficiente de x [b]: x

Digite o valor do coeficiente de x^2 [a dif.0: 6
Digite o valor do coeficiente de x [b]: |
```

Ln: 9 Col: 40

Figura 2b - Saída normal: blindando coeficiente não numérico