Validando Tipagem de Dados

Problemas de tipagem dos dados

Os problemas de tipagem de dados ocorrem quando tentamos realizar uma operação ou ação que não é aplicável ao tipo de dados que estamos trabalhando. Aqui estão alguns exemplos comuns destes erros e como tratá-los:

TypeError: ocorre quando os operandos são incompatíveis, ou seja, quando tentamos somar ou concatenar dois objetos com tipos diferentes, como int e str. Para corrigir isso, precisamos converter o tipo de dados antes de realizar a operação, como:

num = 3

word = "hello"

result = str(num) + word # Converte um dado do tipo numérico para o tipo string antes da concatenação

ValueError: este erro ocorre quando tentamos converter uma string que não pode ser convertida em um número inteiro, como:

num = "three"

int(num) # Lança o erro ValueError

Para corrigir isso, verificamos se a string pode ser convertida em um número inteiro antes de tentar a conversão, como:

num = "3"

if num.isdigit():

result = int(num)

else:

result = "input is not a number"

AttributeError: este erro ocorre quando tentamos aplicar um método a um objeto que não o suporta, como o método 'sort' a um inteiro. Para corrigir isso, precisamos converter o objeto para o tipo apropriado antes de aplicar o método, como:

num = 3

num\_list = [num]

num\_list.sort() # Lança o erro AttributeError

num = 3

num\_list = [num]

num\_list = [str(num) for num in num\_list]

num\_list.sort() # Tem o comportamento esperado

Tipagem de datas

Manipulação de dados de data e hora em Python pode ser uma tarefa desafiadora, especialmente quando lidamos com tipagem errada. Aqui estão alguns exemplos comuns de erros de tipagem em datas e como tratá-los:

TypeError: 'str' object cannot be interpreted as an integer - este erro ocorre quando tentamos usar uma string como entrada para uma função de data que espera um número inteiro, como:

date = "2022-01-01"

datetime.datetime.strptime(date, "%Y-%m-%d").date() # Lança o erro TypeError

Para corrigir isso, precisamos converter a string em um tipo de data apropriado antes de usá-lo como entrada para a função, como:

date = "2022-01-01"

parsed\_date = datetime.datetime.strptime(date, "%Y-%m-%d").date()

ValueError: time data '2022-13-01' does not match format '%Y-%m-%d' - este erro ocorre quando a string fornecida não está no formato esperado pela função, como:

date = "2022-13-01"

datetime.datetime.strptime(date, "%Y-%m-%d").date() # Lança o erro ValueError

Para corrigir isso, precisamos verificar se a string está no formato correto antes de passá-la como entrada para a função, como:

date = "2022-13-01"

if re.match("^\d{4}-\d{2}-\d{2}$", date):

parsed\_date = datetime.datetime.strptime(date, "%Y-%m-%d").date()

else:

parsed\_date = "invalid date format"

TypeError: an integer is required (got type float) - este erro ocorre quando passamos um número flutuante como argumento para uma função que espera um número inteiro, como:

timestamp = 1609459200.0

datetime.datetime.fromtimestamp(timestamp) # Lança o erro TypeError

Para corrigir isso, precisamos arredondar o número flutuante para um inteiro antes de passá-lo como argumento para a função, como:

timestamp = 1609459200.0

datetime.datetime.fromtimestamp(int(timestamp))

OverflowError: date value out of range - este erro ocorre quando passamos uma data inválida para a função de data, como:

date = datetime.date(2022, 2, 29)

datetime.datetime.combine(date, datetime.time.min) # Lança erro OverflowError

Para corrigir isso, precisamos verificar se a data fornecida é válida antes de usá-la como entrada para a função, como:

date = datetime.date(2022, 2, 29)

if date.year % 4 == 0 and (date.year % 100 != 0 or date.year % 400 == 0):

parsed\_date = datetime.datetime.combine(date, datetime.time.min)

else:

parsed\_date = "invalid date"

TypeError: can only concatenate str (not "datetime.date") to str - este erro ocorre quando tentamos concatenar um objeto do tipo data com uma string, como:

date = datetime.date(2022, 1, 1)

"Today is " + date # Lança o erro TypeError

Para corrigir isso, precisamos formatar a data como uma string antes de concatená-la com uma string, como:

date = datetime.date(2022, 1, 1)

"Today is " + date.strftime("%Y-%m-%d")

Esses são apenas alguns exemplos de erros comuns de tipagem de dados em datas. É importante estar ciente desses erros e saber como tratá-los corretamente para garantir que as informações de data e hora sejam processadas de forma confiável em suas aplicações.

Lidando com Fuso Horários

A biblioteca datetime do Python não possui suporte nativo a fusos horários, mas podemos usar a biblioteca pytz para adicionar esse suporte. Aqui estão alguns exemplos de como manipular fusos horários em datas:

Adicionar o fuso horário a uma data sem fuso horário:

import datetime

import pytz

date = datetime.datetime(2022, 1, 1, 0, 0, 0)

utc\_date = pytz.utc.localize(date)

Convertendo uma data de um fuso horário para outro:

import datetime

import pytz

utc\_date = datetime.datetime(2022, 1, 1, 0, 0, 0, tzinfo=pytz.utc)

pst\_date = utc\_date.astimezone(pytz.timezone("America/Los\_Angeles"))

Formatando a saída de uma data com fuso horário:

import datetime

import pytz

pst\_date = datetime.datetime(2022, 1, 1, 0, 0, 0, tzinfo=pytz.timezone("America/Los\_Angeles"))

formatted\_date = pst\_date.strftime("%Y-%m-%d %H:%M:%S %Z")

Verificando se uma data está dentro de uma faixa de horários com fuso horário:

import datetime

import pytz

pst\_date = datetime.datetime(2022, 1, 1, 0, 0, 0, tzinfo=pytz.timezone("America/Los\_Angeles"))

start\_time = datetime.datetime(2022, 1, 1, 9, 0, 0, tzinfo=pytz.timezone("America/Los\_Angeles"))

end\_time = datetime.datetime(2022, 1, 1, 17, 0, 0, tzinfo=pytz.timezone("America/Los\_Angeles"))

if start\_time <= pst\_date <= end\_time:

within\_range = True

else:

within\_range = False

Esses são apenas alguns exemplos de como manipular fusos horários em datas com o pytz. É importante ter em mente que o trabalho com fusos horários pode ser complexo e requer cuidado para evitar erros.

Referências

Tratamento de tipagem errada em Python:

Documentação oficial do Python: https://docs.python.org/3/library/exceptions.html#TypeError

Guia de boas práticas para programação em Python: https://docs.python-guide.org/writing/style/#error-handling

Tratamento de tipagem errada em datas:

Documentação da biblioteca datetime do Python: https://docs.python.org/3/library/datetime.html

Tutorial de manipulação de datas em Python: https://www.w3schools.com/python/python\_datetime.asp

Manipulação de fusos horários em datas:

Biblioteca pytz: https://pythonhosted.org/pytz/