Curso Python Avançado

Tratando e Extendendo Exceções

olist

Sintaxe Exceções

Sintaxe Exceções

Como vimos anteriormente é possível capturar exceções utilizando `try` e `except`.

```
try:
    for i in ['a','b','c']:
        print(i**2)
except:
    print("An error occurred!")
```

Sintaxe Exceções

Python suporta a clausula finally, que deve ser usado sempre que for necessário efetuar o *cleanup* da execução de um try.

Isto porque o bloco definido em finally sempre será executado (após o tratamento da exceção).

```
x = 5
y = 0
try:
    z = x/y
except ZeroDivisionError:
    print("Can't divide by Zero!")
finally:
    print('All Done!')
```

Sintaxe Exceções

Python também suporta a clausula else que define um bloco a ser executado caso nenhuma exceção ocorra durante a execução do bloco try.

```
def ask():
    while True:
        try:
            n = int(input('Input an integer: '))
        except:
            print('An error occurred! Please try again!')
            continue
        else:
            break

    print('Thank you, your number squared is: ',n**2)


ask()
```

Sintaxe Exceções

É possível utilizar `try`, `except`, `else` e `finally` em conjunto:

```
def askint():
    while True:
        try:
            val = int(input("Please enter an integer: "))
        except:
            print("Looks like you did not enter an integer!")
            continue
        else:
            print("Yep that's an integer!")
            break
        finally:
            print("Finally, I executed!")
        print(val)


askint()
```

Sintaxe Exceções

Considerações Importantes

- Se uma exceção ocorre durante a execução da cláusula try, a exceção deve ser tratada por uma cláusula except. Se a exceção não é tratada por uma cláusula except, a exceção é re-lançada depois da cláusula finally ser executada.
- Uma exceção pode ocorrer durante a execução de uma cláusula except ou else. Novamente, a exceção é re-levantada depois que finally é executada.
- Se a instrução try executa um break, continue uma return, a cláusula finally será executada imediatamente após a execução do break, continue ou return.
- Se uma cláusula finally executa uma instrução return, esse instrução return será executada no lugar da instrução return presente na cláusula try.

Exceções Customizadas

Exceções Customizadas

É possível criar exceções próprias criando uma classe sempre de extendendo direta ou indiretamente a classe `Exception`.

O nome da classe deve preferencialmente terminar com o sufixo *"Error"*.

Código a seguir disponível em: https://pastebin.com/5tSNixwN

Exceções Customizadas

```
class Error(Exception):
    """Base class for exceptions in this module."""
    pass

class InputError(Error):
    """Exception raised for errors in the input.

    Attributes:
        expression -- input expression in which the error occurred
        message -- explanation of the error
    """

    def __init__(self, expression, message):
        self.expression = expression
        self.message = message

class TransitionError(Error):
    """Raised when an operation attempts a state transition that's not
    allowed.

    Attributes:
        previous -- state at beginning of transition
        next -- attempted new state
        message -- explanation of why the specific transition is not allowed
    """

    def __init__(self, previous, next, message):
        self.previous = previous
        self.next = next
        self.message = message
```

Boas Práticas com Exceções

Boas Práticas com Exceções

Discussão baseada nas recomendações da PEP 8: https://www.python.org/dev/peps/pep-0008