COM220
Computação Orientada a Objetos I

Aula 11: Exceptions

Falha

- Um programa pode falhar por dois motivos
  - Erros de sintaxe
  - Erros lógicos (Exceptions)

Erros de sintaxe

- if a < 3
  - Esqueci os dois pontos
  - Programa não roda, parser indica o erro, que deve ser corrigido

Exceptions

- Erros que ocorrem em tempo de execução (runtime)
- Diferentes causas
  - Tentar abrir um arquivo que não existe
  - Tentar dividir um número por zero
  - Tentar importar um módulo que não existe
  - ...
- Quanto um erro de runtime ocorre, Python cria um objeto da classe Exception
  - Se não tratado, resulta num erro que na maioria das vezes interrompe a execução do programa

Exceptions

- Python provê um conjunto de built-in Exceptions que são geradas quando ocorre o erro correspondente
- Para ver as built-in exceptions digite
  - print(dir(locals()['__bultins__ ']))

Exceptions

- Alguns exemplos

Exception	Descrição
FloatingPointError	Gerada quando uma operação de ponto flutuante falha
KeybordInterrupt	Gerada quando o usuário pressiona tecla de interrupção (Ctrl + C)
NotImplementedError	Gerada por métodos abstratos
OverflowError	Gerada quando o resultado de uma operação aritmética é muito grande para ser representado
StopIteration	Gerada pela função next() quando não há mais itens para iterar numa coleção

Tratamento de Exceptions

- Quando uma built-in exception ocorre, o interpretador interrompe o processo corrente e passa o controle para o processo chamador até que a Exception seja tratada. Exemplo:
  - Um programa tem uma **função A** que chama uma **função B**, que por sua vez chama uma **função C**
  - Se uma Exception ocorre na **função C** e não é tratada, a exceção é passada para a **função B** e depois para **função A**
    - Se no decorrer dessa sequência não for tratata, o programa falha

Tratamento de Exceptions

- Em Python, o tratamento de Exceptions é feito por meio de cláusulas **try... except**
  - A operação crítica que pode causar a Exception deve ser colocada dentro do **try**
  - O código que trata a Exception deve ser colocado dentro do **except**

Tratamento de Exceptions

```
# importar module sys para pegar o tipo da exception
import sys

lista = ['a', 0, 2]

for elemento in lista:
    try:
        print("O elemento é ", elemento)
        r = 1/int(elemento)
        break
    except:
        print("Oops!", sys.exc_info()[0], "ocorreu")
        print("Próxima entrada")
        print()
print("O número recíproco de", elemento , "é", r)
```

Tratamento de Exceptions

- A execução do código mostra que, quando o código dentro do bloco **try** é executado com sucesso, o bloco **except** é ignorado
  - O **except** serve então para tratar a falha

- Toda exceção em Python herda da classe base **Exception**. Assim, o mesmo código pode ser escrito da seguinte forma

Tratamento de Exceptions

```
lista = ['a', 0, 2]

for elemento in lista:
    try:
        print("O elemento é ", elemento)
        r = 1/int(elemento)
        break
    except Exception as e:
        print("Oops!", e.__class__, "ocorreu")
        print("Próxima entrada")
        print()
print("O número recíproco de", elemento , "é", r)
```

Tratando Exceptions específicas

- No código anterior não foi mencionada nenhuma exception específica
- Isso não é a melhor prática, uma vez que existe um único **except** que trata todos os tipos de erro
- Melhor prática envolve criar um **except** para cada tipo de erro que pode ocorrer
  - Assim, uma cláusula **try** pode ter vários **except**

Tratando Exceptions específicas

```
try:
    # faz algo
    pass

except ValueError:
    # trata ValueError exception
    pass

except (TypeError, ZeroDivisionError):
    # É possível tratar duas ou mais exceções no
    # mesmo except. Aqui duas except são tratadas
    # TypeError and ZeroDivisionError
    pass

except:
    # trata todas as demais exceções
    pass
```

try... except... else

- Suponha que queremos obter o número recíproco apenas de números pares
- Assim:
  - Se número par, calcula número recíproco
  - Se número ímpar, ocorre uma exceção

- Podemos obter esse resultado usando try... except... else

try... except... else

```
# imprime o número recíproco apenas de números pares

try:
    num = int(input("Digite um número: "))
    assert num % 2 == 0
except:
    print("Não é um número par!")
else:
    reciproco = 1/num
    print(reciproco)
```

try... except... finally

- Quando se deseja que um código seja executado após o try independentemente da ocorrência de falha
  - Código dentro do finally executa se houver sucesso ou falha no try

try... except... finally

```
# O try vai gerar uma exception ao tentar escrever num arqui
vo read-only

try:
  f = open("demofile.txt")
  f.write("Lorum Ipsum")
except:
  print("Algo saiu erra na operação de escrita")
finally:
  f.close()

# Finally será executado em qualquer situação, ou seja,
# ocorrendo ou não a exceção
```

Exceptions definidas pelo usuário

- É possível criar Exceptions definindo uma classe que herda da classe Exception
  - A maioria das built-in Exceptions herdam dessa classe

- Vamos criar um programa que ajuda o usuário a adivinhar um número
  - Toda vez que ele chutar um número menor vamos gerar a exception **ValorMenor**
  - Toda vez que ele chutar um número maior vamos gerar a exception **ValorMaior**

Exceptions definidas pelo usuário

```
class ValorMenor(Exception):
    #Gerada quando o valor é menor
    pass

class ValorMaior(Exception):
    #Gerada quando o valor é maior
    pass

# número a ser descoberto
nro = 10
```

Exceptions definidas pelo usuário

```
# usuário tenta adivinhar o número
while True:
    try:
        i_num = int(input("Digite um número: "))
        if i_num < nro:
            raise ValorMenor
        elif i_num > nro:
            raise ValorMaior
        break
    except ValorMenor:
        print("Valor menor, tente novamente!")
        print()
    except ValorMaior:
        print("Valor maior, tente novamente!")
        print()

print("Parabéns! Você descobriu o número.")
```

Exercício

- Considere a criação de um cadastro a partir de uma lista com dados de usuários, a saber: username, email, idade
- As seguintes regras devem ser levadas em consideração:
  - Não é permitido cadastrar um usuário se o seu username já tiver sido usado
  - Menores de 18 anos não podem ser cadastrados
  - Se a idade for inválida (por exemplo, negativa) usuário não pode ser cadastrado
  - O email fornecido deve ser válido, caso contrário o usuário não pode ser cadastrado
- Implemente um programa que usa Exceptions para verificar essas condições, gerando exceções toda vez que uma dessas condições ocorrer