Aula 02

ESTRUTURAS CONDICIONAIS

Estruturas Condicionais e de Repetição

- Controle de Blocos por ***Indentação***
  - Na linguagem Python os blocos são demarcados por espaços, formando uma ***indentação visual***

```
print(“O valor de a é”)
if a == 0:
     print(“zero”)
else:
     print(a)
```

Estruturas Condicionais e de Repetição

- Estrutura condicional ***if***

```
if condição:
      #bloco de código
elif condição:
      #bloco de código
else:
      #bloco de código
```

```
4   v1 = float(input("Valor 1: "))
5   v2 = float(input("Valor 2: "))
6   op = input("Operação {+,-,*,/}: ")
7
8   if op == '+':
9       res = v1+v2
10  elif op == '-':
11      res = v1-v2
12  elif op == '*':
13      res = v1*v2
14  elif op == '/':
15      if v2 != 0:
16          res = v1/v2
17      else:
18          res = "Divisão por zero!"
19  else:
20      res = "Operação inválida!"
21
22  print(res)
```

Estruturas Condicionais e de Repetição

- Estrutura de Repetição ***for***
  - percorre uma sequência previamente conhecida

```
for item in sequência:
      #bloco de código
else:
      #bloco de código
```

  - Uma sequência é uma coleção do tipo: string, listas, tuplas ou buffers.

Exemplo

```python
import os
os.system('cls')

nome = "João da Silva"

for letra in nome:
    print(letra)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
J
o
ã
o

d
a

S
i
l
v
a
```

Exemplo

```python
import os
os.system('cls')

for i in range(5):
    print(i)

for i in range(1,10,2):
    print(i)

for i in range(10,1,-1):
    print(i)

soma = 0
for i in range(5):
    soma += i
else:
    #após terminar a iteração
    print("Soma = %d" %soma)
```

Exemplo

```
4   dados = "abcdefg"
5   for pos, valor in enumerate(dados):
6       print("Posição: {0}, Valor: {1}".format(pos,valor))
7
```

PROBLEMS TERMINAL ••• 2: Python Debu

```
Posição: 0, Valor: a
Posição: 1, Valor: b
Posição: 2, Valor: c
Posição: 3, Valor: d
Posição: 4, Valor: e
Posição: 5, Valor: f
Posição: 6, Valor: g
```

Estruturas Condicionais e de Repetição

- Estrutura de Repetição ***while***
  - Repetição enquanto uma condição é verdadeira

```
while condicao:
        #bloco de código
else:
        #bloco de código
```

Exemplo

```
4    i = 1
5    while i <= 5:
6        print(i)
7        i += 1
```

PROBLEMS TERMINAL ...

```
1
2
3
4
5
```

Estruturas Condicionais e de Repetição

- Controle de repetições
  - ***continue***
    - Iniciar imediatamente a próxima volta do loop
  - ***break***
    - Encerrar imediatamente o loop

ATIVIDADE PRÁTICA

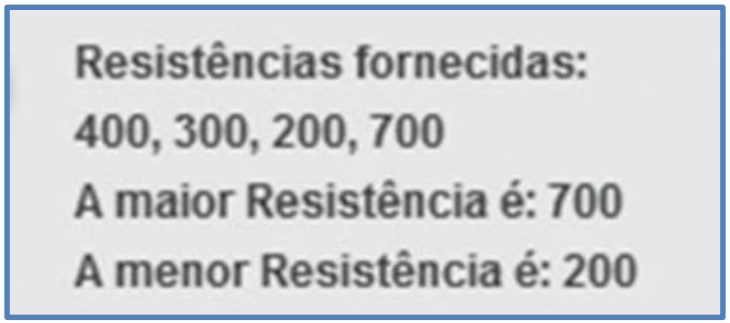
Atividade Prática

- ***Exercício 1***
  - Elabore uma aplicação que receba o nome de um produto e o seu valor. O desconto deve ser calculado de acordo com o valor fornecido conforme a Tabela.
  - Apresente em tela o nome do produto, valor original do produto e o novo valor depois de ser realizado o desconto. Caso o valor digitado seja menor que zero, deve ser emitida uma mensagem de aviso.

Valor (R$)	Desconto (%)
>=50 e <200	5
>=200 e <500	6
>=500 e <1000	7
>=1000	8

Atividade Prática

- ***Exercício 2***
  - Na área da eletrônica, em um circuito em série temos que a resistência equivalente (total) desse circuito é obtida mediante a soma de todas as resistências existentes (RE = r1 + r2 + ... + rn).
  - Faça uma aplicação que receba o valor de quatro resistências ligadas em série, calcule e mostre a resistência equivalente, a maior e a menor resistência.



Atividade Prática

- ***Exercício 3***
  - Faça uma aplicação que apresente em tela a tabuada de qualquer número.
  - O usuário fornece o número desejado e a aplicação apresenta a relação de todos os cálculos de 1 a 10.

Atividade Prática

- ***Exercício 4***
  - Given a year, return the century it is in. The first century spans from the year 1 up to and including the year 100, the second - from the year 101 up to and including the year 200, etc.
  - Example
    - For year = 1905, the output should be centuryFromYear(year) = 20;
    - For year = 1700, the output should be centuryFromYear(year) = 17.

Adapted from <codefighters.com>

TRATAMENTO DE EXCEÇÕES

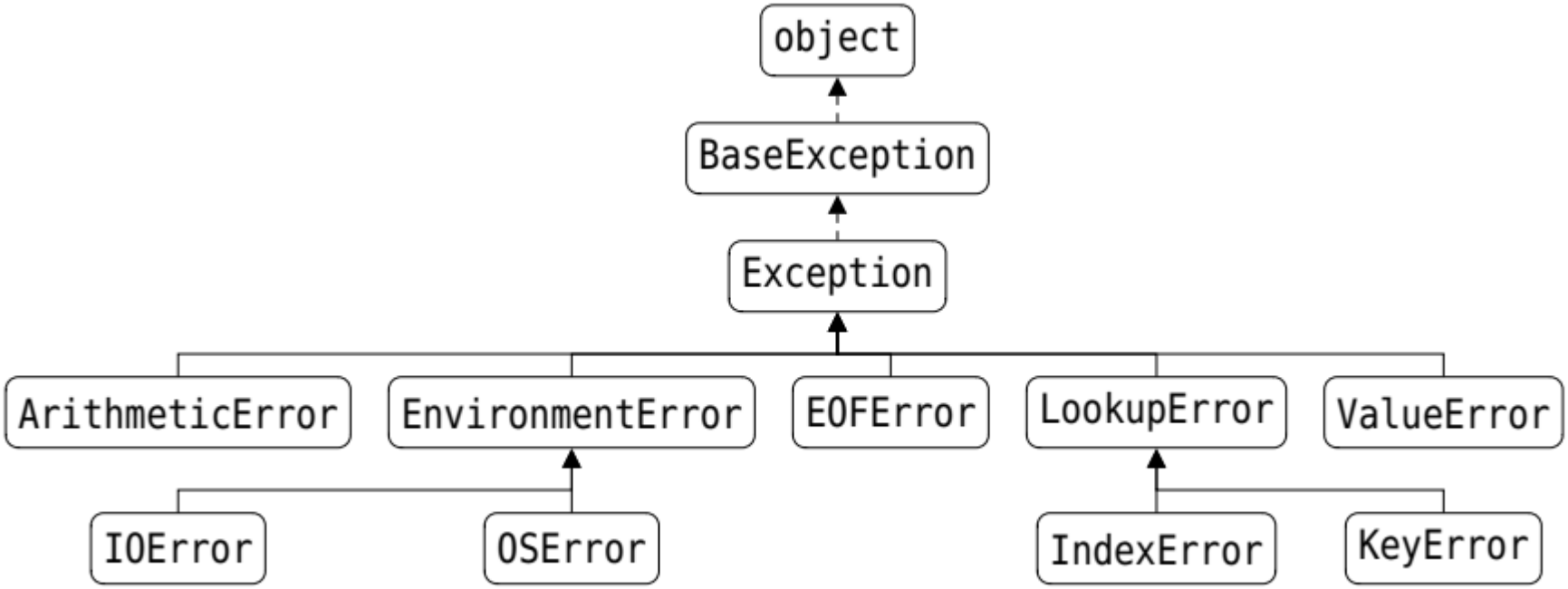
Tratamento de Exceções

- ***Tratamento de Exceções***

```
try:
        #bloco de código
except TipoExcecao as erro:
        #bloco de código
finally:
        #bloco de código
```

Tratamento de Exceções

- ***Tratamento de Exceções***



Tratamento de Exceções

- ***Tratamento de Exceções***

```
5   while True:
6       try:
7           n = int(input("Entre com um número: "))
8       except ValueError:
9           print("O valor informado não é um número")
10
```

PROBLEMS TERMINAL ••• 2: Python Debu

```
Entre com um número: 1
Entre com um número: 2
Entre com um número: a
O valor informado não é um número
Entre com um número: 3
```

Tratamento de Exceções

- ***Tratamento de Exceções***

```
5   while True:
6       try:
7           n = int(input("Entre com um número: "))
8       except ValueError as erro:
9           print(erro)
10
```

PROBLEMS TERMINAL ••• 2: Python Debu

```
Entre com um número: 1
Entre com um número: 2
Entre com um número: a
invalid literal for int() with base 10: 'a'
```

Tratamento de Exceções

- ***Tratamento de Exceções***

```
5    s = "aeiou"
6    try:
7        print(s[5])
8    except IndexError as erro:
9        print("A posição não foi encontrada")
10       print(erro)
11
```

PROBLEMS TERMINAL ••• 2: Python Debu ▾ + 🗑

```
A posição não foi encontrada
string index out of range
```

Tratamento de Exceções

- ***Tratamento de Exceções***

```
4   vet = "12X45"
5
6   for i in range(0,6):
7       try:
8           print("Quadrado: %d" %(int(vet[i])**2))
9       except ValueError:
10          print("Valor inválido na posição: %d" %i)
11      except IndexError:
12          print("Índice %d maior que o tamanho do vetor" %i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Quadrado: 1
Quadrado: 4
Valor inválido na posição: 2
Quadrado: 16
Quadrado: 25
Índice 5 maior que o tamanho do vetor
```

ATIVIDADE PRÁTICA

Atividade Prática

- ***Exercício 1***
  - Elabore uma aplicação que receba o peso e altura de um número indeterminado de pessoas.
  - Utilize tratamento de exceção para garantir que os valores informados são válidos.
  - Após a entrada correta dos dados apresente o IMC.
  - Para cada entrada de dados pergunte ao usuário “Deseja continuar?”

FUNÇÕES

Funções

- Uma função pode ser definida como um bloco de código que contém:
  - Nome
  - Conjunto de parâmetros (ou argumentos)
  - Retorno

```
def nome_função(arg1,arg2,...,argn):
    #bloco de código
    return valor_retorno
```

O retorno é opcional

Exemplo

```python
import os

def limpar_tela():
    os.system('cls')

def ler_numero():
    return int(input("Informe um valor: "))

def soma(n1, n2):
    return n1+n2

def main():
    limpar_tela()
    n1 = ler_numero()
    n2 = ler_numero()
    print(soma(n1,n2))

main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Informe um valor: 9
Informe um valor: 6
15
```

Funções

- Nos parâmetros é possível definir o ***valor padrão*** do argumento

```
1   def exibir_intervalo(ini=0, fim=10, passo=1):
2       for item in range(ini,fim,passo):
3           print(item)
4
5   exibir_intervalo(1,6,2)
6   exibir_intervalo(fim = 5, passo = 2)
```

- Na chamada da função é permitido ***indicar qual o argumento receberá o valor***.

Funções

- Definição de função com ***número indeterminado de parâmetros***

```
1  def exibir_valores(*valores):
2      for valor in valores:
3          print(valor)
4
5  exibir_valores(10,20,30)
```

ATIVIDADE PRÁTICA

Atividade Prática

- ***Exercício 1***
  - Elabore uma aplicação receba um número indeterminado de valores informados pelo usuário.
  - Crie funções para determinar:
    - Quantidade de números pares
    - Quais são os números ímpares
    - O maior número
    - O menor número
    - A média dos números
  - Apresente os resultados na tela.

Atividade Prática

- ***Exercício 2***
  - Elabore uma aplicação capaz de gerar o currículo de uma pessoa em HTML.
  - Os parâmetros para o currículo são: nome, endereço, telefone, e-mail, escolaridade e experiência profissional.
  - Você pode utilizar as tags HTML da sua preferência.
  - Utilize funções para organizar o código fonte da aplicação.