Insper

Design de Software

Aula 3 – Python (condicionais e strings)

2016 – Engenharia

Fabio Ayres <FabioJA@insper.edu.br>
Raul Ikeda <RaulIGS@insper.edu.br>

Objetivos de Aprendizado

- Resolver problemas com operadores relacionais, lógicos e condicionais.
- Fazer operações em *Strings*.
- Laços simples usando *While*.

Expressões booleanas

Dão o resultado **True** ou **False**

Exemplos (no interpretador):

```
1 < 2
=> True
1 == 2
=> False
1 == 1
⇒ True
"i" in "insper"
⇒ True
```

Verdadeiro e Falso

É possível uma variável armazenar um valor de verdadeiro ou falso?

```
resultado = True
```

Como avaliar expressões lógicas

Operador	Operação
==	Igual
!=	Diferente
>	Maior
<	Menor
>=	Maior Igual
<=	Menor Igual

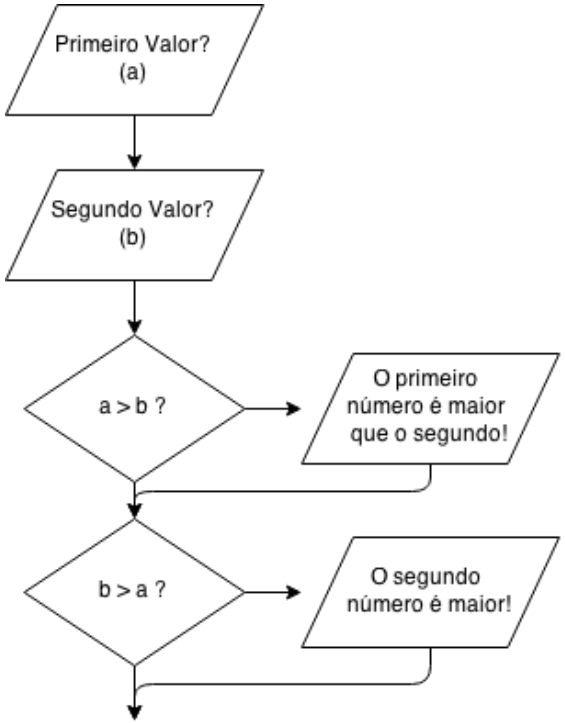
If – execução condicional

Muda o que é executado de acordo com condições

Código

```
a = int(input("Primeiro valor: "))
b = int(input("Segundo valor: "))
if a > b:
    print("O primeiro número é maior que o segundo!")
if b > a:
    print("O segundo número é o maior!”)
```

Flowchart



Execução condicional

Muda o que é executado de acordo com condições

Operadores Lógicos

not – inverte resultado
or – True se antes ou depois True
and – True somente se ambos True

```
if (idade >= 18) and (idade < 21):
    print("Pode beber no Brasil, não nos EUA")
```

Atenção também com a indentação!

Tabela Verdade

p	not p
V	F
F	V

p	q	p and q	p or q
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

Onde p e q são chamadas PREMISSAS.
Ex: p = (idade >= 18) #p só pode ser True (V) ou False (F)

if/else: (Senão)

Fazer um intermediário só if e else agora:

```
if (idade >= 18)  and  (idade < 21):
    print(“Pode beber no Brasil, não nos EUA”)
else:
    print(“Talvez você possa beber, não sei”)
```

Problema

Escreva um programa que pergunta a distância que um passageiro deseja percorrer em km. Calcule o preço da passagem, cobrando R$0.50 por km para viagens de até 200km e R$0.45 para viagens mais longas. (Ex. 4.6 livro Nilo Ney)

Execução condicional (2)

Muda o que é executado de acordo com condições

```
if idade >= 21:
    print("Pode beber nos EUA e no Brasil")
else:
    if idade >= 18:
        print("Pode beber só Brasil")
    else:
        print("Você não pode beber")
```

elif: um else e if juntos?

Indentação (outras linguagens)

Python não usa { } ou Begin End para denotar blocos internos. O nível de indentação que define o bloco a ser executado.

```
if 1 + 1 == 2:
    print("foo”)
    print("bar”)
    x = 42

if 1 + 1 == 2:
    print("foo”); print("bar”); x = 42

# Tudo na mesma linha
if 1 + 1 == 2: print("foo”); print("bar”); x = 42
```

Fonte: http://www.secnetix.de/olli/Python/block_indentation.hawk

Exercícios iniciais

Cada grupo trabalha em um exercício e depois apresenta resultado:

1. Escreva um programa que pergunte a velocidade do carro de um usuário. Caso ultrapasse 80km/h, exiba uma mensagem dizendo que o usuário foi multado. Nesse caso exiba a multa, cobrando R$5,00 por km acima de 80. (Ex. 4.2 livro Nilo Ney)

2. Escreva um programa que pergunte o salário do funcionário e calcule o valor do aumento. Para salários superiores a R$1.250,00, calcule um aumento de 10%. Para inferiores ou iguais, de 15% (Ex. 4.4 livro Nilo Ney)

3. Escreva um programa que leia dois números e que pergunte qual operação você deseja realizar. Você deve poder calcular a soma(+), subtração(-), multiplicação(*) e divisão(/). Exiba o resultado da operação solicitada. (Ex. 4.8 livro Nilo Ney)

4. Escreva um programa para aprovar o empréstimo bancário para compra de uma casa. O programa deve perguntar o valor da casa a comprar, o salário e a quantidade de anos a pagar. O valor da prestação mensal não pode ser superior a 30% do salário. Calcule o valor da prestação como sendo o valor da casa a comprar dividido pelo número de meses a pagar. (Ex. 4.9 livro Nilo Ney)

Imposto de Renda da Pessoa Física (IRPF)

O imposto de renda foi instituído no Brasil no ano de 1923. Desde então, ele vem sendo cobrado anualmente tanto dos cidadãos (pessoas físicas) quanto de outras entidades como as empresas (pessoas jurídicas). Nas nossas aulas, vamos analisar apenas o imposto de renda cobrado das pessoas físicas.

Se você quiser conhecer alguns aspectos históricos da cobrança de impostos no Brasil, como o trecho da lei de 31 de dezembro de 1922 que criou o imposto de renda, consulte:

http://www.receita.fazenda.gov.br/Memoria/irpf/historia/hist1922a1924.asp?ano=1922&origem=1922.

O imposto de renda tem finalidade predominantemente fiscal. Isso significa que se trata de um instrumento de arrecadação de receitas para financiar o Estado, pois sem recursos o Estado não pode exercer suas atribuições mínimas. No entanto, para entender o seu mecanismo de cobrança, é fundamental falarmos de sua finalidade social, relacionada à questão da redistribuição de renda no país.

Levando em consideração a renda de cada pessoa, o sistema de tributação define quem deverá sustentar o financiamento do Estado, e com quanto deve colaborar. Assim, seguindo a lógica de promover a redistribuição de renda, aqueles que têm uma alta renda pagam mais imposto de renda do que os que ganham menos, e aqueles com renda abaixo de um valor mínimo ficam isentos do pagamento.

Observe que, se todos pagassem de imposto, por exemplo, 20% de sua renda, o princípio do "quem ganha mais paga mais" continuaria valendo, já que 20% de R$ 10.000 é mais do que 20% de R$ 2.000. Para aumentar ainda mais a diferença entre os valores pagos por duas pessoas com rendas muito diferentes, o percentual da renda cobrado de imposto depende da faixa de rendimentos de cada pessoa. Os diferentes percentuais cobrados de imposto de renda recebem o nome de *alíquotas*.

Assim, o mecanismo de cobrança do imposto de renda não só faz com que "quem ganhe mais pague mais", mas também garante que "quem ganha mais paga *proporcionalmente* mais".

A explicação acima foi feita de maneira bastante resumida. Ao longo de nossas aulas, vamos analisar com mais detalhes o mecanismo de cobrança do imposto de renda, que faz com que ele seja reconhecido como justo por toda a sociedade.

Existem, ainda, outras finalidades do sistema de tributação de um país, como a política e a econômica. Se você quiser se aprofundar um pouco sobre o assunto, consulte o texto "A Finalidade da Tributação e sua Difusão na Sociedade", de Andréa L. Viol. Ele está disponível em:

http://www.receita.fazenda.gov.br/publico/estudotributarios/eventos/seminarioii/texto02afinalidadedatributacao.pdf

Fonte: Professores Ana Helena, Fabio Orfali e Tadeu da Ponte

Insper

Problema Final (IR)

5) Pergunte ao usuário quanto é seu salário. A seguir imprima quanto ele deve pagar de imposto de renda

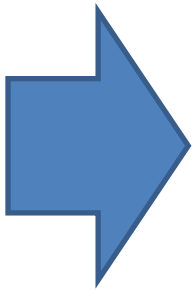
Base de cálculo mensal em R$	Alíquota %	Parcela a deduzir do imposto em R$
Até 1.787,77	-	-
De 1.787,78 até 2.679,29	7,5	134,08
De 2.679,30 até 3.572,43	15,0	335,03
De 3.572,44 até 4.463,81	22,5	602,96
Acima de 4.463,81	27,5	826,15

Fonte: www.receita.fazenda.gov.br

Strings

Cadeias de caracteres

```
palavra = "Insper"

print(len(palavra))

print(palavra[0])

print(palavra[1])

print(palavra[2])

print(palavra[3])

print(palavra[4])

print(palavra[5])
```



Imprime:

```
6
I
n
s
p
e
r
```

Por quê?

Índices

Insper

Mais operações com Strings

```
p = "Insper"
```

`2*p`	InsperInsper
`len(p)`	6
`p[0:3]`	Ins
`p[-1]`	r

Strings – find e replace

```
p = "Insper"
```

```
p.find("er")
```

4

```
p.replace("Ins", "Su")
```

Super

O ponto serve para chamar métodos de strings

Composição de strings

```
nome = "Mateus"
idade = 25
fracao = 1.23456

s = "Meu nome é %s"%nome
print(s)
```

Meu nome é Mateus

Marcador	Tipo
%s	Strings
%d	Inteiros
%f	Floats

OBSOLETO

```
s= "Sou %s de %d, meu float favorito é %3.2f"%(nome,idade,fracao)
print(s)
```

Sou Mateus de 25, meu float favorito é 1.23

Formatação de strings

https://docs.python.org/3.4/tutorial/inputoutput.html

```
>>> import math
>>> print('The value of PI is approximately {0:.3f}.'.format(math.pi))
The value of PI is approximately 3.142.
```

Slicing

http://pythoncentral.io/cutting-and-slicing-strings-in-python/

`p = "Insper"`	Resposta:
`p[1:5]`	`"nspe"`
`p[:3]`	`"Ins"`
`p[3:]`	`"per"`
`p[:3] + p[3:]`	`"Insper"`
`p[0:6:2]`	`"Ise"`
`p[6::-1]`	`"repsnI"`

Atividade em grupos

1. Use find e slicing para recortar uma frase depois da primeira ocorrência de uma dada palavra.

Exemplo:

```
Digite uma frase, por favor:the long and winding road
Digite uma palavra:winding
the long and winding
```

Atividade em grupos

2. Mostre como substituir uma palavra específica em uma frase por sua versão toda em maiúsculas.

Exemplo:

```
Digite uma frase, por favor:here comes the sun
Digite a palavra a substituir:sun
here comes the SUN
```

Atividade em grupos

3. Usando slicing (fatiamento) ou uma função do Python escreva uma frase de trás para frente.

Exemplo:

```
Digite o texto a inverter:
Este é um teste
etset mu é etsE
```

Loops ou laços: *while*

- Um loop extremamente simples

```
1 while True:
2     print("A")
3
```

While – mais um exemplo

- Um exemplo um pouco mais complexo.
  O que ele faz?

```
1 valor = 1
2 while valor!=0:
3     valor = int(input("Digite o valor"))
4     print("Você digitou %d"%valor)
5
```

While – mais um exemplo

- Qual o resultado esperado?

```
1 i = 0
2 while i < 10:
3     print(i, end="")
4     i += 1
5
```

Evita pular para a próxima linha após o if

Para a próxima aula

- Repetições

Capítulo 5 do livro
Introdução à Programação com Python, de Nilo Ney Menezes.
(disponível na biblioteca)

Insper

Insper

www.insper.edu.br

Fun: Easter eggs

>>>import __hello__
Hello world…

>>> from __future__ import braces
  File "<stdin>", line 1
SyntaxError: not a chance

>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!