aprenda PROGRAMAR >>>jogando

INTRODUÇÃO AO PYTHON

SBC Sociedade Brasileira de Computação

CHANCELADO

Campus UFRJ-Macaé
Professor Aloísio Teixeira

UFRJ

APRESENTAÇÃO

Segundo os próprios fundadores da linguagem, “Python is a programming language that lets you work more quickly and integrate your systems more effectively” (“Python é uma linguagem de programação que permite trabalhar mais rapidamente e integrar seus sistemas de maneira mais eficiente”).

O Python é uma linguagem que permite desenvolver aplicações para games, desktops, web e dispositivos móveis, além de poder se comunicar com aplicações desenvolvidas em outras linguagens.

A seguir serão apresentados alguns conceitos básicos e introdutórios para programar em linguagem Python.

1. TIPOS DE DADOS

Tipo de dado é uma forma de classificação de uma informação. As linguagens de programação normalmente trazem implementado o que é chamado de tipos primitivos, isto é, o tipo de dado mais genérico possível. Toda informação é um caractere, seja ele uma letra, um número, um símbolo ou então, um caractere especial. A seguir serão exemplificados três dos mais fundamentais tipos de dados do Python.

1.1 Inteiro

Um valor de tipo inteiro é qualquer número inteiro, isto é, números que não contenham casas decimais.

```
>>> 1
1
>>> 2
2
>>> 1+1
2
>>> 2*3
6
>>> 6/3
2.0
```

Valores do tipo inteiro podem passar por operações básicas como adição (+), subtração (-), multiplicação(*), divisão(/) e potenciação (**).

1.2 Float

Um valor de tipo Float ou Ponto Flutuante é qualquer número decimal, isto é, números que contenham casas decimais.

```
>>> 1.5
1.5
>>> 1.5*2
3.0
>>> 1.5+3
4.5
```

1.3 String

Uma string é um texto, sendo representado entre aspas (“) ou apóstrofo (‘).

```
>>> "Olá, mundo"
'Olá, mundo'
>>> "Olá" + "," + "mundo"
'Olá,mundo'
```

Obs: Strings podem ser concatenadas através do uso do símbolo de adição (+).

2. VARIÁVEIS

Variável é um local onde podemos guardar algum dado ou informação. Podemos usar variáveis para guardar informações como números e palavras.

```
>>> a=2
>>> b=3
>>> a+b
5
```

Variáveis "a" e "b" armazenando dados do tipo inteiro

```
>>> a="Olá"
>>> a + "," + " " + "mundo"
'Olá, mundo'
>>> a
'Olá'
>>> a=a+", mundo!"
>>> a
'Olá, mundo!'
```

Variável "a" armazenando dado do tipo string e tendo seu valor alterado

3. TIPOS DE COMANDOS

3.1 Comando de entrada e saída

O comando de entrada padrão do Python é o input, utilizado para ler entradas de dados. Já o print é o comando de saída padrão, usado para mostrar na tela um valor determinado. O código a seguir exemplifica o uso desses dois comandos:

```
nome=input("Digite seu nome aqui: ")
print("Olá, ", nome)
```

```
Digite seu nome aqui: Beatriz
Olá,  Beatriz
```

3.2 Comando Condicional

Um comando condicional é aquele que permite decidir se um determinado conjunto de comandos deve ou não ser executado, a partir do resultado de uma expressão lógica (condição). Para formular tal expressão lógica utiliza-se os operadores condicionais do Python:

Operador	Tipo	Valor
==	Igualdade	Verifica a igualdade entre dois valores
!=	Igualdade	Verifica a diferença entre dois valores
>	Comparação	Verifica se o valor de a A é maior que B
<	Comparação	Verifica se o valor de B é maior que A
>=	Comparação	Verifica se o valor de A é maior ou igual a B
<=	Comparação	Verifica se o valor de B é maior ou igual a A

A estrutura do comando condicional em Python pode ser simples, composta ou aninhada. O código a seguir receberá dois valores inteiros que serão comparados através de um comando condicional e retornará alguma afirmação a respeito desses valores:

```
x=int(input("Digite um número: "))
y=int(input("Digite um número: "))
```

Estrutura simples:

```
if x==y:
    print("São iguais")
if x!=y:
    print("São diferentes")
```

Estrutura composta:

```
if x==y:
    print("São iguais")
else:
    print("São diferentes")
```

Estrutura aninhada:

```
if x==y:
    print("São iguais")
elif x>y:
    print(x," é maior que ", y)
else:
    print(y," é maior que ", x)
```

ATENÇÃO!!!

Um erro muito comum é confundir o uso das diferentes estrutura dos comandos condicionais, portanto atente-se:

A estrutura simples permite que parte do programa seja executada apenas quando a condição é satisfeita, assim todas as vezes que o programa for rodado as condições serão testadas.

A estrutura composta é usada para simplificar o código, executando os blocos de forma alternativa. Ou seja, se o bloco "if" tiver sua condição satisfeita o bloco "else" não será testado, tornando o programa mais elegante.

A estrutura aninhada é uma contração da estrutura composta, usada para casos em que o número de condições a ser testada é maior, tornando mais claro o tratamento de tais alternativas, através do encadeamento das condições.

Vamos ver um exemplo comum de erro no comando condicional:

```
a=2
b=2
if a == b:
    print("a é igual a b")
if a > b:
    print("a é maior que b")
else:
    print("a é menor que b")
```

Nesse caso, ao rodar o programa, a estrutura simples fará com que ocorra o seguinte problema:

```
a é igual a b
a é menor que b
>>>
```

Para evitar esse erro bastaria usar uma estrutura aninhada:

```
a=2
b=2
if a == b:
    print("a é igual a b")
elif a > b:
    print("a é maior que b")
else:
    print("a é menor que b")
```

Assim , caso a primeira condição seja satisfeita, nenhum dos outros blocos serão testados e, ao rodarmos o programa, teremos o seguinte resultado:

```
a é igual a b
>>>
```

Portanto, para esse caso, a estrutura aninhada soluciona o problema.

3.3 Comando de repetição

Em algumas situações para uma tarefa ser cumprida um certo comando deve ser escrito inúmeras vezes, para deixar o código menor e mais simples de entender podemos usar os comandos de repetição. Loop ou repetição, então, são comandos que se repetem.

Na linguagem Python um comando de repetição comumente usado é o While que faz com que um conjunto de instruções seja executado enquanto uma condição é atendida. Um exemplo da aplicação desse comando será apresentado a seguir:

```
numero=int(input("Digite um número: "))
contador=0
print("Os números pares até ", numero)
while contador<=numero:
    print(contador)
    contador+=2
```

```
Digite um número: 12
Os números pares até  12
0
2
4
6
8
10
12
```

O programa a seguir sorteará um número para que o usuário tente acertar:

```
import random
num1=random.randint(1,10)
num2=int(input("Digite um número entre 1 e 10:"))
if num1==num2:
    print("Você acertou!")
else:
    print("Você errou")
```

Como fazer para repetir a execução até que o usuário acerte?

Uma solução para essa questão é o comando de repetição While True, assim o programa rodará enquanto o critério de parada não for satisfeito, ou seja, enquanto o comando "break" não for chamado:

```
while True:
    import random
    num1=random.randint(1,10)
    num2=int(input("Digite um número entre 1 e 10:"))
    if num1==num2:
        print("Você acertou!")
        break
    else:
        print("Você errou, tente outra vez")
```

```
Digite um número entre 1 e 10:2
Você errou, tente outra vez
Digite um número entre 1 e 10:3
Você errou, tente outra vez
Digite um número entre 1 e 10:1
Você errou, tente outra vez
Digite um número entre 1 e 10:4
Você errou, tente outra vez
Digite um número entre 1 e 10:5
Você acertou!
>>>
```

4. BIBLIOTECAS

A Biblioteca Padrão do Python é uma coleção de módulos acessíveis que simplificam o processo de programação, reduzindo a necessidade de reescrever os comandos mais usados. Essas bibliotecas podem ser usadas quando são chamadas/ importadas. Alguns exemplos de bibliotecas muito utilizadas são: Biblioteca Math, Biblioteca Random e Biblioteca Time.

```
from random import*
numero=randint(0,10)
print("Número aleatório entre 0 e 10: ", numero)

Número aleatório entre 0 e 10:  1
```

Uma lista das Bibliotecas Padrão do Python pode ser encontrada em https://www.python.org/doc/.

5. FUNÇÕES

Funções são blocos de código que realizam tarefas que, normalmente, precisam ser executadas diversas vezes dentro de uma aplicação. Similar ao conceito de função na matemática, o domínio da função seriam os valores que pode-se passar como argumentos e a imagem é o conjunto de possíveis valores de

retorno. Sua estrutura segue um padrão, mas seu método de operação no programa varia de acordo com seu código.

```
def nome_usuario():
    nome=input("Digite seu nome aqui: ")
    return nome
```

Tendo definido uma função a mesma pode ser chamada em qualquer parte do programa. Para isso basta usar o nome da função e fornecer os argumentos necessários, sempre entra parênteses

```
def soma(a,b):
    soma=a+b
    return(soma)

valor1=int(input("Digite um número: "))
valor2=int(input("Digite outro número: "))
resultado=soma(valor1,valor2)
print("A soma dos dois número é ",resultado)
```

No exemplo acima a função "soma" retornará a soma dos dois valores(argumentos) dados pelo usuário, esse valor será guardado pela variável "resultado" e mostrado ao final:

```
Digite um número: 24
Digite outro número: 4
A soma dos dois número é  28
```

6. LISTAS

Uma lista (list) em Python é uma sequência ou coleção ordenada de valores. Cada valor na lista é identificado por um índice. Os valores que formam uma lista são chamados de elementos ou itens.

Listas são similares a strings, que são uma sequência de caracteres, no entanto, diferentemente de strings, os itens de uma lista podem ser de tipos diferentes.

Além disso é possível obter o comprimento (len), concatenar (+), multiplicar por um número (*), verificar a existência de um item (in), obter valor máximo (max), mínimo (min) e soma (sum), adicionar elemento (append), entre outras funções e métodos.

```
numeros=[0,1,2,3,4]
numeros.append(5)
print(numeros)
print(max(numeros))
print(sum(numeros))
```

```
Número aleatório entre 0 e 10:  8
[0, 1, 2, 3, 4, 5]
5
15
```

7. REFERÊNCIAS BIBLIOGRÁFICAS

Python. https://www.python.org/

Pense em Python. https://penseallen.github.io/PensePython2e/