Tipos e formatos de dados (1)

Aula 2

Curso: BIG863 - Basic Python Programming for Ecologists

Professora: Dra. Cecilia F. Fiorini

Supervisor: Prof. Dr. Fernando A. O. Silveira

https://meet.google.com/zdi-ueoz-nsr, 22 de março de 2023

Roteiro*

*Conteúdo adaptado a partir de material desenvolvido pelo Prof. Zanoni Dias e disponível em https://ic.unicamp.br/ mc102.

ECMVS

Imprimindo no Console

- A função print é responsável por imprimir uma mensagem.
- A função print pode ser utilizada para informar o usuário sobre:
  - A resposta de um processamento.
  - O andamento da execução do programa.
  - Comportamentos inesperados do programa.
  - Outros motivos em que o usuário precise ser informado sobre algo.

ECMVS

Imprimindo no Console

- Digitando o seguinte comando:

```
1 print("Hello world!")
```

- Deve aparecer a mensagem:

```
1 Hello world!
```


ECMVS

Imprimindo no Console

- Iremos estudar posteriormente como criar nossas próprias funções, mas agora vamos aprender um pouco mais sobre a função print.
- Como todas as funções, a sintaxe para a função de impressão começa com o nome da função (que neste caso é print), seguida de uma lista de argumentos, incluída entre parênteses.

```
1 print("Argumento 1", "Argumento 2", "Argumento 3")
```

```
1 Argumento 1 Argumento 2 Argumento 3
```


ECMVS

Imprimindo no Console

- Note que, quando informamos mais de um argumento para a função print, eles são automaticamente separados por um espaço.

```
1 print("Hello", "world!")
```

```
1 Hello world!
```

- Podemos modificar isso utilizando o parâmetro sep.

```
1 print("Hello", "world!", sep = "+")
```

Imprimindo no Console

- Os comandos a seguir produzem o mesmo resultado:

```
1 print("Hello world!")
2 print("Hello", "world!")
3 print("Hello", "world!", sep = " ")
```

- Resposta obtida:

```
1 Hello world!
2 Hello world!
3 Hello world!
```


ECMVS

Comentários

- Em Python é possível adicionar um comentário utilizando o caractere #, seguido pelo texto desejado.
- Os comentários não são interpretados pela linguagem, isso significa que todo texto após o caractere # é desconsiderado.
- Exemplo:

```
1 print("Hello world!") # Exemplo de função print
```

- Como resposta para o código acima obtemos apenas:

```
1 Hello World!
```


ECMVS

Comentários

- Vantagens de comentar o seu código:
  - Comentários em trechos mais complexos do código ajudam a explicar o que está sendo realizado em cada passo.
  - Torna mais fácil para outras pessoas que venham a dar manutenção no seu código ou mesmo para você relembrar o que foi feito.

```
1 # Parâmetros importantes da função print
2 # sep: Texto usado na separação dos argumentos recebidos.
3 # end: Texto impresso no final da execução da função.
4 print("MC102", "Unicamp", sep = " - ", end = "!\n")
5 # MC102 - Unicamp!
```


ECMVS

Comentários

- O caractere # é utilizado para comentar uma única linha.
- É possível comentar múltiplas linhas utilizando a sequência de caracteres "' no início e no fim do trecho que se deseja comentar.

```
1 '''
2 Parâmetros importantes da função print
3 sep: Texto usado na separação dos argumentos recebidos.
4 end: Texto impresso no final da execução da função.
5 '''
6 print("MC102", "Unicamp", sep = " - ", end = "!\n")
7 # MC102 - Unicamp!
```


ECMVS

Tipos

- Em Python existem diferentes tipos de dados.
- Podemos ter dados no formato:
  - Numérico.
  - Textual.
  - Lógico.
- Para isso, em Python, temos alguns tipos:
  - **int** Números inteiros (Exemplos: -3, 7, 0, 2020).
  - **float** Números reais (Exemplos: -3.2, 1.5, 1e-8, 3.2e5).
  - **str** Cadeia de caracteres/Strings (Exemplos: "Unicamp"e "MC102").
  - **bool** Valores booleanos: True (Verdadeiro) e False (Falso).

Tipos

- A função type pode ser utilizada para mostrar o tipo de um dado.
- Essa função recebe um argumento que terá o tipo identificado.
- Como resposta, a função informa o tipo do dado fornecido como argumento.
- Exemplo da estrutura da função:

```
1 type(<argumento>)
```


ECMVS

Exemplos de Tipos

```
1 print(type(10))
2 # <class 'int'>
```

```
1 print(type(10.0))
2 # <class 'float'>
```


ECMVS

Exemplos de Tipos

```
1 print(type("10"), type("10.0"))
2 # <class 'str'> <class 'str'>
```

```
1 print(type(True), type(False), type("True"), type("False"))
2 # <class 'bool'> <class 'bool'> <class 'str'> <class 'str'>
```


ECMVS

Variáveis

- Ao escrevermos um código, surge a necessidade de armazenarmos valores de maneira temporária, para isso temos as variáveis.
- Em Python, o caractere = é utilizado para atribuir um valor a uma variável.
- Exemplo:

```
1 pi = 3.1416
2 print(pi)
3 # 3.1416
```


ECMVS

Regras para Nomes de Variáveis

- Nomes de variáveis devem começar com uma letra (maiúscula ou minúscula) ou um sublinhado (_).
- Nomes de variáveis podem conter letras maiúsculas, minúsculas, números ou sublinhado.
- Cuidado: a linguagem Python é case sensitive, ou seja, ela diferencia letras maiúsculas de minúsculas.
- Por exemplo, as variáveis c1 e C1 são consideradas diferentes:

```
1 c1 = 0
2 C1 = "1"
3 print(c1, type(c1), C1, type(C1))
4 # 0 <class 'int'> 1 <class 'str'>
```


ECMVS

Operadores aritméticos

+	**Adição** Realiza a soma de ambos operandos.
-	**Subtração** Realiza a subtração de ambos operandos.
*	**Multiplicação** Realiza a multiplicação de ambos operandos.
/	**Divisão** Realiza a Divisão de ambos operandos.
//	**Divisão inteira** Retorna o quociente inteiro de uma divisão, ignorando o resto.
%	**Módulo** Retorna o resto da divisão de ambos operandos.
**	**Exponenciação** Retorna o resultado da elevação da potência pelo outro.

ECMVS

Operadores aritméticos - Ordem de precedência

- Precedência é a ordem na qual os operadores serão avaliados quando o programa for executado. Em Python, os operadores são avaliados na seguinte ordem:

Precedência

exponenciação > multiplicação = divisão > módulo > adição = subtração

- Podemos controlar a ordem com que as expressões são avaliadas com o uso de parênteses.
- Procure usar sempre parênteses em expressões para deixar claro em qual ordem as mesmas devem ser avaliadas.

ECMVS

Operadores de comparação

`==` **Igual a** Verifica se um valor é igual ao outro.

`!=` **Diferente de** Verifica se um valor é diferente ao outro.

`>` **Maior que** Verifica se um valor é maior que outro.

`>=` **Maior ou igual** Verifica se um valor é maior ou igual ao outro.

`<` **Menor que** Verifica se um valor é menor que outro.

`<=` **Menor ou igual** Verifica se um valor é menor ou igual ao outro.

ECMVS

Operadores de lógicos

and	Retorna True se ambas as afirmações forem verdadeiras
or	Retorna True se uma das afirmações for verdadeira
not	Retorna Falso se o resultado for verdadeiro

ECMVS

Operadores lógicos - Ordem de precedência

- A ordem de precedência dos operadores lógicos é a seguinte:

Precedência

not > and > or

- Podemos controlar a ordem com que as expressões são avaliadas com o uso de parênteses.
- Procure usar sempre parênteses em expressões para deixar claro em qual ordem as mesmas devem ser avaliadas.

ECMVS

Exemplos com operadores lógicos

```
1 (3 < 4) and ("banana" > "abacaxi")
2 # True
```

```
1 (4 == 4.0) and (4 == "4")
2 # False
```

```
1 (3 >= 4) and ("casa" > "peixe")
2 # False
```


ECMVS

Operadores de identidade

is	Retorna True se ambas as variáveis são o mesmo objeto.
is not	Retorna True se ambas as variáveis não forem o mesmo objeto.

ECMVS

Conversões de Tipos

- Alguns tipos de dados permitem que o seu valor seja convertido para outro tipo (cast).
- Para isso, podemos usar as seguintes funções:
  - **int()** converte o valor para o tipo int (número inteiro).
  - **float()** converte o valor para o tipo float (número real).
  - **str()** converte o valor para o tipo str (string).
  - **bool()** converte o valor para o tipo bool (booleano).

ECMVS

Exemplos de Conversões de Tipos

- Convertendo uma string para um número inteiro:

```
1  a = "45"
2  b = int(a)
3  a
4  # '45'
5  type(a)
6  # <class 'str'>
7  b
8  # 45
9  type(b)
10 # <class 'int'>
```


ECMVS

Exemplos de Conversões de Tipos

- Valores booleanos podem ser convertidos para números:

```
1 int(True)
2 # 1
3 int(False)
4 # 0
5 float(True)
6 # 1.0
7 float(False)
8 # 0.0
```


ECMVS

Recebendo Dados do Usuário

- A função input é responsável por receber dados do usuário.
- O usuário deve escrever algo e pressionar a tecla <enter>.
- Normalmente, armazenamos o valor lido em uma variável.
- A função input obtém os dados fornecidos pelo console no formato de string (str).
- Devemos fazer uma conversão dos dados se quisermos trabalhar com números.

ECMVS

Exemplos de Entrada de Dados

- Sintaxe da função input:

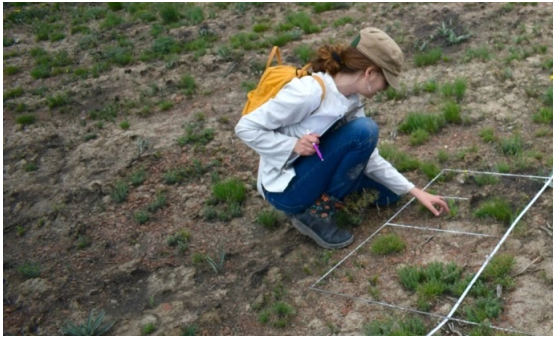
```
1 x = input("Mensagem opcional")
```

- Armazenando os valores lidos nas variáveis a e b:

```
1 a = input("Digite um valor para a variável a: ")
2 b = input("Digite um valor para a variável b: ")
3 print(int(a) + float(b))
```


ECMVS

Parcelas de Estudo



UFMG_Ecopyhton_aula02

ECMVS