Preparação OBI 2019

01/10/2018

Entrada & Saída de dados

Entrada de dados entendemos como os dados que vem de fora do nosso algoritmos que são inseridos para serem processados ou utilizados em alguma rotina com finalidade específica.

A saída de dados é a resposta que nosso programa dá durante a execução ou ao término da mesma.

Quando nos referimos a entrada e saída padrão utilizaremos o console/terminal para ler os dados (entrada padrão) e para a saída do programa (saída padrão)

Entrada & Saída de dados

Para quem conhece alguma outra linguagem já deve ter ouvido falar de STDIN e STDOUT

Linguagens de alto nível tornam mais simples a implementação de comandos de entrada e saída.

Vamos ver alguns exemplos utilizando python...

Entrada & Saída de dados

```
#coding: utf-8

input("Escreva entre aspas alguma informação a ser impressa no Console ")
```

Vamos testar esse comando de duas formas em python: primeiro abra um terminal e digite python3, ao fazer isso vocês irão abrir um console interativo do python3, uma forma de testar comandos e realizar rápidas execuções.

Agora que vocês testaram no console interativo vamos fazer um pouco diferente… Salvem em um arquivo o mesmo comando digitado no console interativo, utilizando um editor de texto da preferência de vocês (atom, kate, vim, etc.), feito isso vamos executar o arquivo… Vá até onde o arquivo está salvo e digite: “**python3 entrada.py**” note que eu salvei o arquivo com o nome entrada.py e estou na mesma pasta em que o arquivo está salvo.

Entrada & Saída de dados

Ainda estão curiosos sobre a entrada de dados? Vamos explorar um pouco mais esse recurso

```
#coding: utf-8

num = input("Digite um número:")
print(num)
```

Tente a execução da mesma forma que foi explicado anteriormente. Agora temos um comando a mais **print**, lembra da saída padrão que foi comentado antes? Então... Esse comando é responsável pela saída padrão

Entrada & Saída de dados

Vamos treinar mais um pouco agora...

```
#coding: utf-8


login = input("Login:")
senha = input("Senha:")

print("O usuário informado foi: %s, e a senha digitada foi: %s" %(login, senha))
```

Variáveis

Variáveis são um dos recursos mais básicos das linguagens de programação. Utilizadas para armazenar valores em memória, elas nos permitem gravar e ler esses dados com facilidade a partir de um nome definido por nós. Vamos ver alguns exemplos através do console interativo:

```
mensagem = 'Exemplo de mensagem!'
n = 25
pi = 3.141592653589931
```

Variáveis

Você notou alguma coisa diferente em relação a outras linguagens de programação? Onde é que está o tipo da variável você deve ter pensado.

Pois é... uma ótima pergunta. Não se assuste, existe um tipo, mas a linguagem abstrai ele para você.

Lembra as variáveis que definimos no *slide* anterior? Vamos colocar o nome de cada variável dentro de uma função chamada ***type()***

type(mensagem***)***

O que aconteceu? Depois de ter testado todas as variáveis crie suas próprias variáveis e teste essa função.

Variáveis

Você acha que pode sair colocando qualquer coisa como nome de variável? Espera só um instante e veja que tem alguns nomes que não devem ser utilizados de forma alguma no seu programa....

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Tipos de dados

O tipo é uma forma de classificar as informação. As linguagens de programação normalmente trazem implementado o que é chamado de tipos primitivos, isto é, o tipo de dado mais genérico possível.

Conseguem imaginar alguma utilidade para isso? E porque tem isso em python se não precisa declarar nada?

Tipos de dados - String

Na programação String representa um conjunto de caracteres disposto numa determinada ordem. A partir de agora, todas as vezes em que falarmos o termo String, estaremos nos referindo a um conjunto de caracteres.

nome=”Adenisvaldo Pereira”

rg=”9.564.988-54”

Tipos de dados - numérico

Um segundo tipo de informação são os dados compostos por caracteres numéricos (algarismo). Os números são divididos em 2 partes:

inteiros - chamados de integer ou int

ponto flutuante - chamado de float ou double

Funções de conversão

Antes de falar sobre isso temos que entender uma coisa, o que são funções... Funções são blocos de código que já vem embutida na linguagem que estamos usando ou programadas por nós, utilizando as funções **input()** e **print()** para ler e exibir dados, respectivamente.

O bacana de utilizar funções é que elas podem nos retornar valores ou não, no caso do **input()**, será retornado o que for digitado na entrada padrão, só que com um detalhe, será tudo String. Lembra do tipo String?

E o que acontece se eu precisar que seja um número?

Funções de conversão

A solução é simples... Vamos converter os valores =)

Ah,e se eu quiser um número e fizer a conversão, só que ao invés de digitar um número a pessoa digitar um nome. Isso tem como resolver, mas não se preocupe por enquanto.

Vamos ver como são as funções de conversão:

int(valor**)**

float(valor**)**

str(valor**)**

Estruturas Condicionais

Sabemos bastante coisa já, só que está tudo muito fácil ainda. Que tal complicar um pouco?

Vamos fazer condições...

Sempre que precisamos fazer comparações utilizamos as estruturas condicionais que a linguagem nos oferece mais algum operador lógico ou aritmético.

Estruturas Condicionais

Você já deve ter visto alguns desses operadores

Operador	Tipo	Valor
==	Igualdade	verifica a igualdade entre dois valores
!=	Igualdade	verifica se dois valores são diferentes
>	Comparação	verifica se A é maior do que B
<	Comparação	verifica se A é menor do que B
>=	Comparação	verifica se A é maior ou igual a B
<=	Comparação	verifica se A é menor ou igual a B
in	Sequência	verifica se o conjunto A está contido em um conjunto

Estruturas Condicionais

Estrutura Condicional Simples

```
if soma > 0:
    print("Maior que Zero.")
```

Condição mais simples e com apenas uma verificação

Estruturas Condicionais

A **Estrutura Condicional Composta executa um comando quando a condição for verdadeira e outra condição quando for falsa**. Vamos melhorar o nosso exemplo anterior, agora teremos que mostrar a mensagem "Menor que Zero" caso o resultado da soma seja menor que zero, como podemos ver abaixo:

```
if soma > 0:
    print("Maior que Zero.")
else:
    print("Menor que Zero.")
```

Estruturas Condicionais

Estruturas Condicionais Aninhadas são várias condições em cascatas, ou seja, um IF dentro de outro IF. Uma outra estrutura de aninhamento é como pode-se notar abaixo. Incrementando o nosso exemplo, agora teremos que exibir uma mensagem caso o valor seja igual a Zero.

```
if soma > 0:
    print("Maior que Zero.")
elif soma = 0:
    print("Igual a Zero.")
else:
    print("Menor que Zero.")
```

Estruturas Condicionais - operadores lógicos

Os operadores lógicos por sua vez, permite-nos unir 2 expressões ligando-as com os conectivos lógicos matemáticos que são, o conectivo **E (and)** e o conectivo **OU (or)**.

VALOR LÓGICO

O valor lógico é um tipo de dado binário, isto é, assume um valor dentre duas opções: verdadeiro ou falso.

```
if idade >= 18 and idade <= 70 :
    print("É necessário votar!")
```

Exercícios

Álbum de Fotos

https://www.t-obi.com/problem/show/2/20

Avião

https://www.t-obi.com/problem/show/2/89

Exercícios

Bocha

https://www.t-obi.com/problem/show/2/12

Campeonato

https://www.t-obi.com/problem/show/2/2

Referências

- http://excript.com/python/entrada-dados-python.html
- https://www.devmedia.com.br/python-trabalhando-com-variaveis/38644
- http://excript.com/python/tipos-de-dados-python.html
- https://www.programiz.com/python-programming/statement-indentation-comments
- https://www.devmedia.com.br/aprendendo-a-programar-em-python-estruturas-condicionais-if/17358
- http://excript.com/python/operadores-logicos-python.html