Instruções básicas de utilização

Tipos de dados:

Cython é uma linguagem fortemente tipada, por conta disto, não é possível alterar o tipo de uma variável já declarada.

Abaixo estão listados todos os tipos de dados existentes na linguagem**:**

number -> Este tipo de dado é utilizado para números
str -> Este tipo de dado é utilizado para strings
bool -> Este tipo de dado é utilizado para booleanos

Definir variáveis:

Abaixo estão definidas as 2 formas possíveis de se definir uma variável.

1° Forma – Definição e atribuição de valor

- **number abc = 1; ->** Cria uma variável abc do tipo number e atribui a ela o valor 1
- **str test = "oi"; ->** Cria uma variável test do tipo str e atribui a ela o valor "oi"
- **bool test = True; ->** Cria uma variável test do tipo bool e atribui a ela o valor True

2° Forma – Definição

- **number abc; ->** Cria uma variável abc do tipo number
- **str test ; ->** Cria uma variável test do tipo str
- **bool test; ->** Cria uma variável test do tipo bool

Fim da linha

Cada linha deve terminar com uma instrução de END_COMMAND, representada na linguagem pelo caractere “**;**”

Operadores

Operadores aritméticos:

A linguagem conta com todos os operadores matemáticos básicos, como soma, subtração, divisão e multiplicação, com estes operadores você é capaz de realizar operações matemáticas, tal como incrementar valores de variáveis, subtrair valores e com este poder você pode até mesmo desenvolver uma calculadora.

Operadores:

- **+ ->** Operador de soma
- **- ->** Operador de subtração
- **/ ->** Operador de divisão
- **->** Operador de multiplicação

Exemplo:

- **number test = 1; ->** O valor da variavel test inicialmente é 1
- **Test = test + 1; ->** O valor da variavel test agora após a soma é 2

Operadores Lógicos:

A linguagem conta com todos os operadores lógicos (and e or).

Operadores:

- **& ->** Operador de AND
- **| ->** Operador de OR

Exemplo:

```
if fisrt_number > second_number & second_number > third_number{
   printf(fisrt_number);
   printf(second_number);
   printf(third_number);
};
```

Operadores de Comparação:

Abaixo estão instruções para o uso de operadores de comparação na linguagem:

- **Operador ==**

  Utilizando este operador é possível verificar a igualdade de dois dados.

  Sintaxe:

  <dado> == <dado>

  Exemplo:

  bool test = (1 == 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação entre 1 e 1

- **Operador >**

  Utilizando este operador é possível verificar se um dado é maior do que o outro.

  Sintaxe:

  <dado> > <dado>

  Exemplo:

  bool test = (1 > 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação 1 maior que 1

- **Operador <**

  Utilizando este operador é possível verificar se um dado é menor do que o outro.

  Sintaxe:

  <dado> < <dado>

  Exemplo:

  bool test = (1 < 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação 1 menor que 1

- **Operador >=**

  Utilizando este operador é possível verificar se um dado é maior ou igual do que o outro.

  Sintaxe:

  <dado> >= <dado>

  Exemplo:

  bool test = (1 > 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação 1 maior ou igual a

- **Operador <=**

  Utilizando este operador é possível verificar se um dado é menor ou igual do que o outro.

  Sintaxe:

  <dado> <= <dado>

  Exemplo:

  bool test = (1 > 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação 1 menor ou igual a 1

- **Operador !=**

  Utilizando este operador é possível verificar a igualdade de dois dados.

  Sintaxe:

  <dado> != <dado>

  Exemplo:

  bool test = (1 == 1) -> Cria uma variável test e atribui a ela o valor resultante da comparação entre 1 e 1

Loops:

A linguagem possui a estrutura iterativa WHILE, que segue exemplificada abaixo:

Sintaxe:

```
while <condicional> {
    <código a ser executado>
}
```

Condicionais:

Abaixo estão instruções para o uso de condicionais na linguagem:

- **Condicional IF**

  Sintaxe:

  ```
  if <condicional> {
      <código>
  };
  ```

- **Condicional ELIF**

Se for a primeira instrução elif deve existir uma condicional if acima desta, caso contrário deve ser seguida de outra condicional elif.

Sintaxe:

```
elif <condicional> {
    <código>
};
```

- **Condicional ELSE**

Para ser utilizada uma condicional else deve existir previamente uma condição if ou elif.

Sintaxe:

```
else {
    <código>
}
```

Funções Nativas:

As funções nativas existentes na linguagem são inputf e printf, para se utilizar o método inputf, a variável que guardará o dado inserido pelo usuário, deve ser criada e atribuído o valor do inputf na mesma linha.

Sintaxe:

- **inputf() ->** Método responsável por receber um input do usuário

- **printf("<texto>" | <variável>) ->** Método responsável por exibir na tela o valor da variável ou conteúdo passado como parâmetro para a função

Exemplo:

- **number test = inputf(); ->** Recebe os dados inseridos pelo usuário e os armazena na variável test.

- **printf("Esse projeto vale nota 10"); ->** Exibe na tela a mensagem "Esse projeto vale nota 10"

Exemplo de código:

O código abaixo é a implementação de uma calculadora matemática básica na linguagem Cython:

```
printf("Digite o 1 numero:");
number first_number = inputf();
printf("Digite a operacao:");
str operation = inputf();
printf("Digite o 2 numero:");
number second_number = inputf();
number result;

if operation == "+"{
   result = first_number + second_number;
   printf(result);
};
elif operation == "-"{
   result = first_number - second_number;
   printf(result);
};
elif operation == "/"{
   result = first_number / second_number;
   printf(result);
};
elif operation == "*"{
   result = first_number * second_number;
   printf(result);
};
else{
   printf("Voce fez coisa errada");
};
```