MC102 - Algoritmos e Programação de Computadores

MC102 Oferecimento anterior

Tarefa de laboratório 00

O objetivo deste exercício é propiciar a familiarização com o interpretador Python e a ferramenta de submissão e testes automáticos SuSy. Teste as opções corretas e todos os erros documentados. Esta experiência poderá facilitar muito o seu processo de desenvolvimento e depuração das próximas tarefas.

Em muitos tutoriais, o primeiro programa ensinado é o que escreve a mensagem *Hello, world!*. Adotaremos esta abordagem e escreveremos:

Oi, mundo!

Veja o código abaixo:

```
print("Oi, mundo!")
```

Testes com a shell

Abra um terminal e o programa python3:

```
$ python3
Python 3.6.6 (default, Jul 19 2018, 14:25:17)
[GCC 8.1.1 20180712 (Red Hat 8.1.1-5)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

A partir deste ponto, você pode testar comandos em Python de forma interativa. Vamos lá!

```
>>> print("Oi, mundo!")
Oi, mundo!
```

Muitas vezes não escrevemos exatamente como o interpretador esperava. São erros de sintaxe ou de definição de nomes. Veja os exemplos abaixo:

- **Falta de parênteses**

  ```
  >>> print "Oi, mundo!"
    File "", line 1
      print "Oi, mundo!"
                        ^
  SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Oi, mundo!")?
  ```

  Neste caso, o interpretador teve facilidade para identificar nosso erro e nos apresentou uma mensagem bem amigável, nos direcionando para a solução.

- **Função com nome inválido**

  Outro erro comum é não digitarmos corretamente o nome de uma função.

  ```
  >>> prit ("Oi, mundo!")
  Traceback (most recent call last):
    File "", line 1, in
  ```

```
NameError: name 'prit' is not defined
>>>
```

Às vezes o nome está correto, mas o interpretador precisa de ajuda para encontrar a função. Estudaremos isso mais para a frente na disciplina.

- **String sem delimitador final**

  Às vezes, a mensagem pode ser um pouco mais difícil de entender. Abaixo, o interpretador está indicando que a linha acabou (EOL = end of line) enquanto tentava processar a string.

```
>>> print("Oi, mundo!)
  File "", line 1
    print("Oi, mundo!)
                      ^
SyntaxError: EOL while scanning string literal
```

- **String sem delimitador inicial**

  Um erro semelhante leva a uma mensagem bem diferente. Neste caso, como não havia aspas demarcando o início da string, o interpretador tentou avaliar o seu conteúdo como código Python e indicou que a sintaxe está inválida.

```
>>> print(Oi, mundo!")
  File "", line 1
    print(Oi, mundo!")
                    ^
SyntaxError: invalid syntax
```

Testes com arquivo lab00.py

Outra forma de usar a linguagem Python é criarmos um arquivo com o programa para execução posterior. Abra um editor de textos como o `gedit` e crie um arquivo chamado `lab00.py` com o conteúdo abaixo:

```
print("Oi, mundo!")
```

e depois execute da seguinte forma:

```
$ python3 lab00.py
Oi, mundo!
```

Altere o arquivo introduzindo erros de sintaxe semelhantes aos anteriores e observe as mensagens retornadas.

Testes com o SuSy

No SuSy, para cada tarefa, criamos um conjunto de testes com arquivos de entrada `arq1.in, arq2.in, arq3.in,..., arqN.in`. A cada um deles corresponde um arquivo de saída `arq1.res, arq2.res, arq3.res,..., arqN.res`. O seu programa passará em um teste `i` apenas se a saída for exatamente igual a do arquivo de resposta `arq<i>.res`. Este é o desafio!

No caso desta tarefa, teremos apenas um arquivo de teste `arq1.in`, sem conteúdo. O arquivo de saída `arq1.res` contém `Oi, mundo!`.

Se você submeter arquivos que não respeitam a sintaxe de Python como os exemplificados na seção anterior, receberá respostas parecidas às documentadas. Agora, observe os comandos abaixo:

```
print("Hello, world")
print("Hallo, Welt!")
print("Oi, mundo!!!")
print("Oiiii, mundooo!")
```

Todos executam sem emissão de mensagem de erro pelo interpretador e cumprem a missão de cumprimentar nosso mundo. No entanto, não seguem exatamente a especificação inicial do enunciado e implicarão em um

resultado **incorreto** para o SuSy. Observe os exemplos abaixo:

- **String com diferenças visíveis**

  Código submetido:

  ```
  print("Oi, mundo!!!!")
  ```

  Resultado:

  ```
  Teste 1: resultado incorreto

      1c1
      < Oi, mundo!!!!
      ---
      > Oi, mundo!
  ```

  O texto precedido por `<` é a primeira e única linha de saída do seu programa e o texto precedido por `>` é a linha da saída esperada. Fica fácil identificar a diferença!

- **String com espaço extra ao final**

  Código submetido:

  ```
  print("Oi, mundo! ")
  ```

  Resultado:

  ```
  Teste 1: resultado incorreto

      1c1
      < Oi, mundo!
      ---
      > Oi, mundo!
  ```

  A saída do seu programa parece idêntica à esperada, mas, se selecionarmos o texto do resultado, a diferença ficará evidente:

  ```
  Teste 1: resultado incorreto

      1c1
      < Oi, mundo!
      ---
      > Oi, mundo!
  ```

- **Linha extra com caracteres visíveis**

  Suponha que o nosso código tentou ser um pouco mais amigável com o mundo:

  Código submetido:

  ```
  print("Oi, mundo!")
  print("Tudo bem?")
  ```

  Resultado:

  ```
  Teste 1: resultado incorreto

      2d1
      < Tudo bem?
  ```

  Se o seu arquivo apresentar a saída esperada seguida de linha(s) extra(s), apenas esta diferença será exibida no resultado.

- **Linha extra vazia**

  Código submetido:

```
print("Oi, mundo!")
print()
```

Resultado:

```
Teste 1: resultado incorreto

    2d1
    <
```

Aqui, apenas o caractere `<` é exibido, visto que a última linha está vazia.

Orientações para submissão

Veja aqui a página de submissão da tarefa. O arquivo a ser submetido deve se chamar **`lab00.py`**.

Para ganhar nota integral nesta tarefa você deve submeter pelo menos um código incorreto e um correto. A última submissão deve ser correta.

O limite máximo será de 10 submissões.

Se você não estiver inscrito corretamente, envie email para islene@ic.unicamp.br.

O peso desta tarefa é 1.

O prazo final para submissão é 26/08/2018.
