Algoritmos e Programação de Computadores II - COM120 - Turma 003
Página inicial
Avisos
Cronograma
Atividades
Fóruns
Colaborate
Calendário Lives
Notas
Menu das Semanas
Semana 1
Semana 2
Semana 3
Semana 4
Semana 5
Semana 6
Semana 7
Semana 8
Orientações Gerais para a Avaliação
Exame
Documentos e informações gerais
Gabaritos
Facilitadores da disciplina
Repositório de REA's
Página Inicial!

Revisar envio do teste: Semana 1 - Atividade Avaliativa

Usuário	LIZIS BIANCA DA SILVA SANTOS
Curso	Algoritmos e Programação de Computadores II - COM120 - Turma 003
Teste	Semana 1 - Atividade Avaliativa
Iniciado	23/04/23 18:42
Enviado	23/04/23 18:49
Data de vencimento	28/04/23 05:00
Status	Completada
Resultado da tentativa	10 em 10 pontos
Tempo decorrido	7 minutos
Instruções	Olá, alunos e alunas! Esta atividade possui múltipla escolha. Para respondê-la: 1. Selecione, com o mouse, a alternativa que você considerar correta; 2. Repare que, ao selecionar uma alternativa, as seleções anteriores são desmarcadas; 3. Após selecionar a resposta correta em todas as questões, vá até o fim da página e clique em "Enviar teste". Prontos! Sua atividade já está registrada no AVA.
Resultados exibidos	Todas as respostas, Respostas enviadas, Respostas corretas, Comentários, Perguntas respondidas incorretamente

Pergunta 1

2,5 em 2,5 pontos

Uma variável em Python é criada a partir do momento em que um nome é utilizado pela primeira vez do lado esquerdo de um comando de atribuição, como em: >>> a = 1 (essa operação cria uma variável chamada "a" e lhe atribui o valor 1). Em variáveis inteiras, considere a situação: *a* recebe 3. *b* recebe *a*. Em seguida, *a* recebe 6.

Assinale a alternativa que contém os valores adequados para as variáveis *a* e *b* conforme a situação citada no enunciado.

Resposta Selecionada: b. >>> a = 6 e >>> b = 3.

Respostas:
- a. >>> a = 3 e >>> b = 6.
- b. >>> a = 6 e >>> b = 3.
- c. >>> a = 6 e >>> b = 0.
- d. >>> a = 0 e >>> b = 3.
- e. >>> a = 3 e >>> b = 3.

Comentário da resposta: **JUSTIFICATIVA**

Nas atribuições de variáveis do tipo inteiro, como no caso em que "a recebe 3" (>>> a = 3), o *a* aponta para o objeto 3 criado na memória do tipo inteiro, e fazendo "*b* recebe *a*" (>>> b = a) o que acontece é que *b* recebe o valor que está armazenado na variável *a*. Como o valor 3 já estava criado (na memória alocada), a variável *b* aponta para o mesmo endereço de memória que vai ser o objeto 3, portanto, tanto *a* quanto *b* vão apontar para o mesmo endereço de memória. E se "a recebe 6" (>>> a = 6) o que vai acontecer é que a variável *a* vai apontar para um novo objeto com o valor 6 e o 3 (que estava apontando para 3) permanece apontando para o objeto com valor 3. Logo, haverão 2 objetos criados na memória com duas variáveis apontando para cada objeto. Sempre que for preciso utilizar a variável *a* para alguma operação, será resgatado o objeto para o qual ela aponta no momento da realização da operação, desse modo, >>> a = 6 e >>> b = 3.

Pergunta 2

2,5 em 2,5 pontos

A leitura e a gravação de arquivos em Python possibilitam a gravação dos dados em disco, por isso existem duas formas de se trabalhar com arquivos, uma que permite abri-los e editá-los em qualquer editor de texto (arquivos de texto) e outra que permite manipular imagens, sons, vídeos etc. (modo binário).

Com base nas informações apresentadas, identifique se são verdadeiras (V) ou falsas (F) as afirmativas a seguir.

I. Usando Python, ao abrir um arquivo de texto para realizar sua leitura, o que ocorre é: a leitura de seus *bytes*, sua decodificação e sua interpretação conforme uma tabela de caracteres para que gere como resultado um objeto *string*.
II. Usando Python, ao abrir um arquivo binário para realizar sua leitura, o que ocorre é: a leitura de seus *bytes*, que logo após são transferidos para memória, sendo interpretados e decodificados.
III. Usando Python, ao realizar uma gravação em um arquivo de texto, o que ocorre é: a codificação dos caracteres de uma *string* para que sejam transformados em *bytes*, possibilitando sua gravação em disco.

Assinale a alternativa que apresenta a sequência correta.

Resposta Selecionada: e. V - F - V.

Respostas:
- a. F - V - V.
- b. F - F - V.
- c. V - F - F.
- d. V - V - F.
- e. V - F - V.

Comentário da resposta: **JUSTIFICATIVA**

A afirmativa I é verdadeira, pois um arquivo de texto tem seus *bytes* lidos, decodificados e interpretados (conforme uma tabela de caracteres) para gerar uma *string*. Um arquivo de texto pode ser aberto no Python com o uso da função open cuja sintaxe padrão é "open(*nome*, *modo*, *buffering*)", o "*nome*" refere-se ao arquivo que será aberto, o "*modo*" contempla a forma de abertura do arquivo e "*buffering*" é a quantidade de *bytes* reservados na memória para a abertura do arquivo, mas isso é opcional.
A afirmativa II é falsa, porque um arquivo no modo binário em Python é lido e levado à memória sem interpretação ou decodificação, concentram os *bytes* em uma forma bruta, por isso é preciso que o programador crie o programa de uma forma adequada com o objetivo de interpretar esses *bytes*.
A afirmativa III é verdadeira, pois, na gravação de um arquivo de texto em Python, os caracteres da *string* são codificados e transformados em *bytes* para que sejam gravados em disco. Assim, para escrever o texto em um arquivo em Python, é preciso que ajustar a forma de abertura do arquivo e fazer uso do método "write()" do objeto *file*, que recebe como argumento o texto que será inserido no arquivo, da seguinte forma: "arquivo.write(texto_a_ser_escrito)".

Pergunta 3

2,5 em 2,5 pontos

Um programa, ao ser implementado, pode conter erros que devem ser corrigidos, por isso a necessidade de um depurador, que é usado para testar o programa implementado identificando seus defeitos, portanto, a depuração consiste na identificação e na redução de erros em programas. Para realizar a correção de forma ágil e eficiente, é preciso inicialmente saber identificar o tipo de erro ocorrido.

Avalie as afirmações a seguir em relação aos tipos de erros e relacione-as adequadamente aos termos às quais se referem.

1 - Erros de sintaxe (*syntax errors*).
2 - Erros de execução (*runtime errors*).
3 - Erros de semântica (*semantic errors*).

I - Este tipo de erro ocorre no momento da execução do programa e indica que houve um erro de lógica.
II - Este tipo de erro ocorre no momento em que o Python está traduzindo o código-fonte do programa para código executável (*byte code*).
III - Este tipo de erro ocorre a partir do momento em que o programa não apresenta mais erros durante sua execução e não apresenta erros de sintaxe.

Assinale a alternativa que correlaciona adequadamente os dois grupos de informação.

Resposta Selecionada: a. 1-II; 2-I; 3-III.

Respostas:
- a. 1-II; 2-I; 3-III.
- b. 1-I; 2-II; 3-III.
- c. 1-I; 2-III; 3-II.
- d. 1-III; 2-II; 3-I.
- e. 1-III; 2-I; 3-II.

Comentário da resposta: **JUSTIFICATIVA**

A sentença I se enquadra no conceito 2, pois os erros de execução são gerados no momento da execução do programa e, normalmente, indicam que há um erro de lógica, com mensagens que informam um erro de execução indicando o ponto do programa onde houve o erro (ponto), o erro de lógica causador do erro pode se localizar em um trecho do código bem distante desse ponto) e informando, também, quais funções estavam sendo executadas, como no caso da recursão infinita que origina erro de execução "*maximum recursion depth exceeded*".
A sentença II se enquadra no conceito 1, pois os erros de sintaxe ocorrem quando o Python está traduzindo o código-fonte do programa em código executável (*byte code*) e normalmente é informado que tem algo escrito sintaticamente errado no programa, como no caso em que não escrever : (dois pontos) no final da linha com um *def* para uma mensagem de "*SyntaxError: invalid syntax*".
A sentença III se enquadra no conceito 3, pois os erros de semântica que acontecem a partir do momento em que não se apresentam mais erros de sintaxe e erros de execução, e com a execução completa do programa ele acaba não produzindo o resultado esperado, isso indica que houve um erro de lógica, como no caso em que uma expressão não é avaliada na ordem em que se deseja, gerando o resultado errado. Um exemplo é o cálculo da expressão a / (b * c) que no programa foi escrito na forma a / b * c. Essa expressão acaba sendo avaliada como (a / b) * c devido à ordem de precedência dos operadores.

Pergunta 4

2,5 em 2,5 pontos

Um arquivo é uma área no disco em que é possível gravar e ler dados, sendo que para ler um arquivo em Python é necessário que ele seja aberto em modo de leitura. A linguagem Python oferece três métodos que permitem ler o conteúdo de um arquivo, que são: read(), readline() e readlines(). Eles permitem ler um arquivo chamado "professores.txt" que contém os nomes de alguns professores e que se localiza no mesmo diretório do script Python.

Avalie os códigos a seguir em relação aos métodos em Python que possibilitam a leitura de um arquivo e relacione-os adequadamente aos resultados de impressão às quais se referem.

1

```
arquivoEscola = open('professores.txt')
nomes = arquivoEscola.read()
print(nomes)
```

Fonte: Elaborada pela autora.

2

```
arquivoEscola = open('professores.txt')
nomes = arquivoEscola.readlines()
print(nomes)
```

Fonte: Elaborada pela autora.

3

```
arquivoEscola = open('professores.txt')
nomes = arquivoEscola.readline()
print(nomes)
```

Fonte: Elaborada pela autora.

I

```
Readlines: ['Jorge Nunes\n', 'Patricia Queiroz\n', 'Manuela Santos Silva\n', 'Camila Rodrigues Alves\n']
```

Fonte: Elaborada pela autora.

II

```
Readline: ['Jorge Nunes']
```

Fonte: Elaborada pela autora.

III

```
Read: ['Jorge Nunes Patricia Queiroz Manuela Santos Silva Camila Rodrigues Alves']
```

Fonte: Elaborada pela autora.

Assinale a alternativa que correlaciona adequadamente os dois grupos de informação.

Resposta Selecionada: a. 1-III; 2-I; 3-II.

Respostas:
- a. 1-III; 2-I; 3-II.
- b. 1-I; 2-III; 3-II.
- c. 1-II; 2-I; 3-III.
- d. 1-I; 2-II; 3-III.
- e. 1-III; 2-II; 3-I.

Comentário da resposta: **JUSTIFICATIVA**

A sentença I se enquadra no conceito 2, pois *Readlines* cria uma lista contendo *strings* em que cada uma representa o nome de um professor do arquivo "professores.txt", portanto cada linha é uma *string* que termina no caractere '\n', e no caso em que o arquivo de texto possuir por exemplo 10.000 linhas, com o uso de "readlines()" será criada (impressa) uma lista com 10.000 de elementos.
A sentença II se enquadra no conceito 3, já que o método "readline()" que está no singular retorna somente uma linha, ou seja, faz a leitura do arquivo e retorna na forma de uma *string*, se o for especificado, ele lê no máximo n *bytes*, porém não lê mais de uma linha, mesmo se n exceder o comprimento da linha.
A sentença III se enquadra no conceito 1, pois ao ler um arquivo com *Read*, o método pega todo o conteúdo do arquivo de texto e o distribui em apenas uma *string*, com todos os nomes juntos.

Quinta-feira, 15 de Agosto de 2024 19h27min08s BRT

OK