INF101 – Introdução à Programação II
Roteiro de Prática: 13 de maio de 2021

Introdução

Vamos considerar, na prática de hoje, o mesmo problema da aula anterior, qual seja, a elaboração de um *horário escolar.* Só que desta vez vamos organizar o programa utilizando o conceito de *classe.* A arquitetura do programa será simplificada ao máximo, estando longe, portanto, de uma utilização real. O objetivo da prática é estritamente acadêmico, ou seja, didático. Para tanto, vamos criar uma classe para modelar o setor de registro escolar de uma faculdade ou universidade. Em uma situação real, teríamos que criar várias outras classes para representar os alunos, os professores, as disciplinas, as salas de aula etc. Vamos considerar os seguintes atributos para a classe `RegistroEscolar`:

- `periodo` – uma 2-tupla contendo (ano, período do ano)
- `disciplinas` – uma lista para conter todas as disciplinas do período
- `matriculas` – um dicionário para conter o conjunto das disciplinas em que cada aluno se matriculou no período; a chave será o nome do aluno
- `horario` – uma lista contendo as sessões possíveis para as disciplinas não conflitantes

Os dados para povoar as estruturas de dados dos atributos serão lidos de dois arquivos: um contendo, por linha, as disciplinas a ser oferecidas no período e o outro contendo, por linha, o nome de um aluno e as disciplinas em que ele se matriculou no período. Para elaborar o programa, siga as instruções abaixo.

Instruções

1. Faça o *download* do esqueleto do programa que se encontra no Sistema de Entrega do LBI. Abra o IDLE e mude o nome do arquivo-fonte para `p14.py`. Não se esqueça de salvá-lo de tempos em tempos, porque pode ocorrer falha de energia elétrica ou do próprio sistema computacional durante a aula prática.
2. Complete os comentários obrigatórios (nome, matrícula, data de criação, data de atualização e uma breve descrição sobre o que o programa faz).
3. O programa está estruturado em: a classe `RegistroEscolar` e a função `main`.
4. Nesta aula, você completará o código-fonte que já veio pronto no esqueleto preenchendo a declaração da classe `RegistroEscolar` com seus atributos definidos na Introdução e os métodos: `set_disciplinas`, `set_matriculas` e `set_horario`. É claro que, em uma situação real, esta classe deveria ter muitos outros métodos.
5. A função `main()` faz o seguinte: primeiro, decide quais serão os nomes externos dos arquivos de entrada, se serão usados os nomes *default* ou os nomes passados como argumentos pela linha de comando. Depois, instancia um objeto da classe `RegistroEscolar` denominado apenas `res`. Observe que, o construtor `__init__` da classe, além do parâmetro `self`, deverá ter os parâmetros `ano` e `periodo` do ano letivo. Em seguida, chama o método `set_disciplinas` sobre o objeto `res` passando como argumento o nome externo do arquivo contendo as disciplinas a ser oferecidas no período em questão. Depois, chama o método `set_matriculas` sobre o objeto `res` passando agora o nome externo do arquivo contendo as matrículas dos alunos no período considerado. Em sequência, chama o método `set_horario`, sem nenhum argumento extra, para elaborar o horário. Para finalizar, ela imprime as sessões possíveis produzidas pelo método `set_horario` e que estão armazenadas no atributo `horario` do objeto `res`. A `main` já está pronta no esqueleto. Favor não a modificar.

6. Declare a classe `RegistroEscolar`. O seu construtor `__init__` deve definir os atributos descritos acima na Introdução e os parâmetros `ano` e `periodo`. Não se esqueça do prefixo `self.` para os identificadores dos atributos.
7. O método `set_disciplinas(self, nomearq)` lê o arquivo de entrada cujo nome externo é passado pelo parâmetro `nomearq`. Este método deve usar os blocos `try-except` para efetuar uma manipulação segura do arquivo. Além disso, ele deve armazenar, no atributo `disciplinas`, uma lista contendo os dados lidos por linha do arquivo de entrada. O método não deve retornar nada, porque ele produz o efeito necessário no próprio atributo `disciplinas` da classe. Não se esqueça do prefixo `self.` para indicar que `disciplinas` é um atributo de um objeto da classe.
8. O método `set_matriculas(self, nomearq)` é semelhante ao anterior (item 7 acima). Ele recebe o argumento com o nome externo do arquivo de entrada por meio do parâmetro `nomearq`. Lê o arquivo de forma segura dentro de um bloco `try-except`. Cada linha do arquivo é lida e dela são extraídos o nome do aluno e as respectivas disciplinas em que ele se matriculou. Em cada linha, os dados estão separados por vírgula, `','`. Os dados são armazenados no dicionário `matriculas` em que as chaves são os nomes dos alunos e os valores associados, o conjunto das disciplinas matriculadas pelo respectivo aluno. O método não precisa retornar nada. (**Obs.:** Não é necessário fazer aqui, agora, a validação das disciplinas matriculadas face à lista das disciplinas oferecidas, porém, em uma situação de fato prática, seria necessário validar os dados das matrículas). Não se esqueça do prefixo `self.` para manipular o atributo `matriculas` da classe.
9. O método `set_horario(self)` elabora o horário de acordo com a heurística dada em aula teórica. Implemente a função passando o código em Python dado na aula teórica para cá, sob a forma de função/método. O método não precisa retornar nada, porque ele age diretamente sobre os atributos da classe. Novamente, não se esqueça do prefixo `self.` ao manipular os atributos da classe.
10. Teste seu programa com os dados dos arquivos fornecidos, `disciplinas.txt` e `matriculas.txt`. Veja como deve ser a saída impressa nos Exemplos de Teste do Programa dados abaixo.
11. Se seu programa entrar em *laço infinito* ou travar por alguma razão, digite `CTRL-C` na janela do *Python Shell* para interromper a execução do programa.
12. Teste o seu programa também com os arquivos extras fornecidos, `disciplinas2.txt` e `matriculas2.txt`. Passe esses nomes de arquivos como argumentos da linha de comando. O IDLE permite fazer isso selecionando, para a execução do arquivo-fonte, a opção `'Run... Customized'` no menu `'Run'`, ou por meio do atalho `Shift-F5`.

☞ Não se esqueça de preencher o cabeçalho do código fonte com seus dados, as datas e uma breve descrição do programa.

Após certificar-se de que seu programa esteja correto, envie o arquivo com o código fonte (`p14.py`) através do Sistema de Entrega do LBI.

Exemplo 1 de Teste do Programa (com os arquivos *default*)

```
Sessões para o período 2020/2:
  0:  ['INF100', 'INF101']
  1:  ['FIS203', 'INF103', 'MAT140']
  2:  ['MAT141', 'MAT144']
  3:  ['LET100']
```

Exemplo 2 de Teste do Programa (com os arquivos disciplinas2.txt **e** matriculas2.txt**)**

```
Sessões para o período 2020/2:
  0:  ['INF100', 'INF101']
  1:  ['INF103']
  2:  ['MAT140']
```