Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Data: 26 de Agosto de 2017
Professora: Elloá B. Guedes
Disciplina: Fundamentos Teóricos da Computação

Projeto Prático I
Expressões Regulares em Python

1 Conhecendo a plataforma Run.Codes

A disciplina de *Fundamentos Teóricos da Computação* possui projetos práticos em sua avaliação, os quais possuem **caráter individual** e **obrigatório** e que serão executados por intermédio da plataforma *Run.Codes*.

O primeiro passo a ser realizado é o cadastro na plataforma. Acesse o site run.codes e, utilizando o seu **nome completo** e **e-mail institucional**. Após esta etapa, procure a disciplina **Fundamentos Teóricos da Computação** (ESTECP006) e cadastre-se na sua turma utilizando o código **62UK**. Todos os alunos devem obrigatoriamente se cadastrarem até o dia **30/08/2017**. O não cadastro, o cadastro em turma incorreta ou a não realização dos exercícios nos prazos estabelecidos culminará em nota zero.

Para quem não conhece, o run.codes é uma plataforma automatizada para testes de entrada e saída. Cada aluno submete o seu código escrito na linguagem Python e este código é submetido a um conjunto de testes, previamente escritos e cadastrados pela professora e monitora. A nota do aluno é individual e obtida de maneira automática, correspondendo ao percentual de acertos nos testes. Por exemplo, se há 15 casos de testes cadastrados e o aluno acertou 12, a nota obtida é 8.

A partir do momento em que o exercício inicia até o momento do seu encerramento, o aluno pode submeter o código para a plataforma quantas vezes quiser, sem que isso afete a nota final. Por exemplo, se um aluno submeteu 10 vezes e acertou 12/15 casos de teste, a nota será a mesma de um outro aluno que acertou 12/15 mas que submeteu 300 vezes.

Uma outra regra a ser considerada na plataforma é que a última versão do código é sempre a que será considerada. Imagine que o aluno João Última Hora está enlouquecidamente programando o exercício e já tem 14/15 casos corretos mas, nos segundos finais do prazo limite submete alterações em seu código e cai para 8/15 acertos. Se o sistema encerrar, a versão 8/15 será a considerada para avaliação. Se João Última Hora tivesse aproveitado melhor o tempo e começado a resolver o exercício desde o momento em queficou disponível na plataforma, não teria passado esse sufoco e ficado com uma nota final tão ruim. Todos podemos aprender com o drama de João Última Hora e evitar tais problemas.

Algumas dicas finais para você ter um bom desempenho nos problemas práticos são:

1. Considere que seu programa recebe uma entrada de cada vez;
2. Efetue testes em seu programa antes de submetê-lo ao run.codes. É uma forma simples de conhecer como seu programa se comporta e uma oportunidade de acertar mais testes logo de primeira;
3. Aproveite o tempo;
4. Há boatos de que você não deve deixar seu computador saber quando você está com pressa!

2 Apresentação do Problema

O GitHub é uma plataforma de hospedagem de código-fonte com controle de versões e que usa Git. Diversos desenvolvedores ao redor do mundo usam esta plataforma para controlarem e compartilharem os códigos-fonte dos projetos que desenvolvem.

A cada segundo, uma grande quantidade de usuários do mundo inteiro interage com a plataforma. Mas, para evitar problemas, as transações precisam ser validadas antes de serem executadas no servidor. O seu objetivo consiste em usar expressões regulares para validar as transações, que são compostas dos seguintes elementos:

1. **Autor**. É um id de usuário na plataforma GitHub. Os ids de usuário são formados por caracteres alfabéticos e numéricos, mas sempre iniciados por caracteres alfabéticos. A quantidade de caracteres numéricos em um id não pode superar o número de caracteres alfabéticos. São exemplos de ids de autores válidos: elloa, jessicaLopes, ell04, el104h, elloaBGuedes, Elloa;
2. **Senha**. Para dificultar a ação de hackers, as senhas no GitHub são compostas de 4 pares de dígitos que podem ser letras de A até F e números de 0 a 9 e que são separados por ponto. Para aumentar a segurança, duas letras não podem aparecer juntas e nenhum par de números pode ter dígitos iguais. Senhas válidas: $03.A5.2B.F8$, 14.35.28.92, etc.
3. **IP do autor**. Reflete o IP do dispositivo utilizado pelo autor para realizar a transação. É um número de 32 bits dividido em 4 octetos de bytes representados no formato decimal. São exemplos de IPs válidos: 192.168.1.2, 127.0.0.1, 255.255.255.255;
4. **E-mail do autor**. Seguindo um formato de email simplificado, em que o início do endereço é feito com uma letra, há um arroba, e pelo menos um ponto após o arroba. São exemplos de e-mails válidos: ebgcosta@uea.edu.br, elloa.uea@gmail.com, c4rl0s@teste.com.au.;

5. **Tipo da transação**. O tipo da transação pode ser uma das opções a seguir: `pull`, `push`, `stash`, `fork` e `pop`;

6. **Repositório**. Indica o nome do repositório vinculado à transação solicitada. Os nomes do repositório são escritos em *snake_case* utilizando apenas letras minúsculas do alfabeto. São exemplos de nomes de repositórios válidos: projeto_ftc, pp1_ftc_uea, projetoftc;

7. **Hash**. A cada transação está associado o resultado de um hash md5. Você deve checar se o hash da transação é válido, sendo composto por 32 dígitos formados por números de 0 a 9 ou caracteres de a até f em minúsculo neste cenário.

Os elementos que compõem uma transação são apresentados em uma única linha conforme a ordem especificada, separados por espaços em branco. Uma transação é válida quando todos os seus elementos são caracterizados de maneira adequada, na ordem especificada, como apresentado anteriormente. Não precisa fazer checagens se a conta e o repositório realmente existem no GitHub, se o e-mail está num servidor válido ou afins, assuma que isso irá compor outro módulo de validação e que uma outra pessoa está responsável por implementar isto. De maneira similar, não preocupe-se em saber o que gerou o hash md5, apenas verifique se o hash que aparece na entrada é válido. A equipe de criptografia está empenhada nestas outras preocupações.

De maneira resumida, a entrada do seu problema é uma string contendo uma transação. A saída é a palavra "True" quando a transação é válida e "False" em caso contrário.

Para resolver o problema em questão, você deve utilizar a linguagem de programação Python 3 e obrigatoriamente fazer uso de expressões regulares. Soluções que não fizerem uso de expressões regulares serão anuladas.

3 Exemplos de Entradas e Saídas

Entrada	**Saída**
elloa A6.B7.C8.F9 192.168.11.0 elloa@github.com push dominar_o_mundo e25df22a1b41ec5248f5af0d8fb1c2dd	True
3ll04 A6.B7.C8.FF 266.168.11.0 3lloa@github.com ploft exemplo-projeto-errado e25df22a1b41ec5248f5af0d8fb1c2dj	False

4 Observações Importantes

- Lembre-se, a entrada de dados é feita via `input` e a saída via `print`;

- Atenha-se exatamente ao padrão de entrada e saída fornecidos nos exemplos. Qualquer mensagem adicional na entrada ou na saída de dados pode culminar em incorretude;
- A cada execução do programa será fornecida apenas uma entrada, cujo resultado deve ser exibido ao final do processamento;
- Na construção do seu programa você deve usar apenas os conceitos aprendidos em sala de aula. Respostas que utilizem bibliotecas prontas não serão consideradas;
- Em caso de plágio, todos os envolvidos receberão nota zero!
- Na execução do seu programa no *run.codes*, existem casos de testes que vão além dos exemplos mostrados a seguir. Esses casos de teste não serão revelados. Pense em exemplos de entradas e saídas que podem acontecer e melhore o seu código para capturá-las.

5 Prazos Importantes

- **Início**. 30/08/2017 às 8h (horário do servidor)
- **Encerramento**. 01/09/2017 às 23h55min (horário do servidor)

6 Links Úteis

- https://developers.google.com/edu/python/regular-expressions
- http://goo.gl/SWwe4R
- https://www.debuggex.com/cheatsheet/regex/python
- http://www.miraclesalad.com/webtools/md5.php