# Formação Desenvolvedor Python Ultima.School

**Módulo 3 -** Controle de versão do Git

**Semana 2 -** Operações comuns do Git

**Instruções**: Os exercícios foram criados para ajudar a fixar o conteúdo e praticar tudo que foi visto durante a semana. É muito importante que eles sejam implementados para fixar o conteúdo. As soluções podem ser o mais simples possível.

**Tutores**: Caso você precise de ajuda para completar os exercícios, os tutores estão à disposição no chat do telegram!

**Feedback**: Os tutores vão revisar as atividades enviadas pela plataforma, na página da atividade; e vão dar feedback!

**Soluções/Gabarito**: os alunos que tenham enviado as atividades, e já tenham recebido feedback; podem pedir acesso às soluções/gabarito dos exercícios se tiverem interesse. Este será liberado e estará disponível somente semanas após a conclusão do módulo em questão.

### **Exercícios**

1. Existem várias situações no fluxo de trabalho com Git onde podemos encontrar conflitos durante o *merge* de branchs. Na grande maioria dos casos, poucas alterações no código podem resolver estes conflitos, sendo então necessário atualizar a branch em questão. Qual a sequência de passos deve ser seguida após um comando **git merge** falhar devido a conflitos?

**RESPOSTA:** Utilizaremos o comando git status para sabermos qual o arquivo que tem algum conflito. Assim, vamos abrir o arquivo no nosso editor e encontrar onde está o conflito, decidiremos qual branch será mantida, usaremos o comando git add para nossas alterações e subiremos com o comando git commit -m

1. Luizinho estava trabalhando em uma melhoria em seu código utilizando a branch *feature/ABC123*  quando colegas de trabalho lhe informaram de um bug em produção que precisa ser corrigido o quanto antes. O bug em questão estava no arquivo *hello.py* na branch *main* do repositório. Qual sequência de comandos git, Luizinho deve executar para corrigir o bug e voltar ao seu trabalho? Lembre-se que a empresa em que Luizinho trabalha segue à risca o GitFlow.  
   **RESPOSTA:** Usará o comando git stach para salvar suas modificações no arquivo atual, usará o comando git checkout hello.py , realizar as modificações necessárias, após, git commit -m. Depois, git stash apply para voltar onde estava e continuar.
2. No fluxo de trabalho de software a principal forma de sincronização de códigos é por meio de ferramentas de versionamento. A mais popular delas é amplamente difundida e é conhecida como git. Tendo em vista os temas abordados em aula, descreva a funcionalidade e exemplifique os comando **git pull** e **git push**   
   **RESPOSTA:** git pull, baixa as modificações do repositório remoto, para o repositório local, utilizado para atualizar seu arquivos junto ao do repositório dos demais parceiros.  
   Git push, é o contrário, ele sobe os arquivos do repositório local para o remoto, serve para subir os arquivos que você modificou para ficar disponível para os demais.
3. Realize uma pesquisa sobre as diversas plataformas de versionamento com git, escolha duas e realize um comparativo com o GitHub avaliando as semelhanças, diferenças, vantagens e desvantagens para o desenvolvimento de software.

**RESPOSTA:** GitLab: Tem o mesmo propósito que o GitHub, porém é de plataforma opensource, todos podem contribuir. A principal vantagem do GitLab é poder desenvolver de forma online, na própria ferramenta.  
Bitbucket: O bitbucket tem o propósito voltado a pequenas equipes, com orçamento menor. Principal vantagem é o fato de possuir suporte para git e mercurial

1. Criei um programa que calcule o peso ideal de uma pessoa. Para isso utilize as fórmulas da tabela:

| Para Homens: (72.7 \* altura) – 58 |
| --- |
| Para Mulheres: (62.1 \* altura) – 44.7 |

Sua aplicação deve identificar se a pessoa é Homem ou Mulher e então fazer o cálculo apropriado. Deve ser impresso se a pessoa é homem ou mulher, juntamente com o peso ideal calculado.

1. Um nutricionista está precisando de uma ajuda para calcular o IMC de seus pacientes. Para calcular o IMC ele passou a seguinte fórmula: *IMC = peso / ( altura )²*. Crie um programa que faça o cálculo do IMC de uma pessoa (ele deve ser impresso na tela) e classifique o IMC dessa pessoa de acordo com a tabela (também deverá ser impresso):

| **Valor do IMC** | **Classificação** |
| --- | --- |
| Abaixo de 18,5 | Pessoa abaixo do peso |
| Entre 18,5 e 25 | Pessoa com peso normal |
| Entre 25 e 30 | Pessoa acima do peso |
| Acima de 30 | Pessoa obesa |

1. Uma loja de móveis está com dificuldades de calcular algumas condições de pagamento de seus móveis. Eles possuem algumas regras e o seu trabalho é implementar uma aplicação que faça os cálculos de acordo com as regras:

| **Regras** |
| --- |
| À vista em dinheiro, recebe 15% de desconto |
| À vista no cartão de crédito, recebe 10% de desconto |
| Em duas vezes, preço normal de etiqueta sem juros |
| Mais de duas vezes, preço normal de etiqueta mais juros de 10% |

O programa deve ter uma variável com o valor de etiqueta do produto, uma variável com forma de pagamento e uma com o preço final após a aplicação de uma das regras.

1. Alguns alunos de uma universidade criaram uma “criptografia” para se comunicarem entre eles durante o tempo que estão longe da universidade. Essa criptografia é baseada em códigos que, quando convertidos, formam as letras de uma palavra. Esses números são informados em uma lista de caracteres e são separados pela string ‘-1’ conforme o exemplo abaixo:  
     
   Nessa sequência teríamos o número 79 representando um caractere e o número 73 representando outro caractere da mensagem. Para saber a letra será necessário percorrer essa lista e ir concatenando os números antes de identificar um separador (‘-1’) para então identificar qual letra o código numérico representa. Por exemplo:
2. A primeira iteração da lista será lido o primeiro número ‘7’, adicionamos ele em uma variável (sugestão de nome: codigo\_letra),
3. Na segunda interação será lido o número ‘9’ que será concatenado na mesma variável variável (usando o += com strings)
4. Na terceira iteração iremos identificar que é o caractere que separa as letras da mensagem, converter o código numérico para uma letra usando o código: ***palavra += chr(int(codigo\_letra))*** (esse código utiliza funções, iremos estudar mais a respeito ao longo do curso). Após converter a variável deve ser limpa para que possamos continuar a ler as demais letras.
5. Será repetido todos os passos acima para a segunda letra

Para o exemplo acima, a primeira letra é o caractere “O” e a segunda letra será o caractere “I” formando a palavra “OI”.  
Sua tarefa será criar uma aplicação que percorra a sequência:  
***mensagem\_criptografada = ['8', '5', '-1', '7', '6', '-1', '8', '4', '-1', '7', '3', '-1', '7', '7', '-1', '6', '5', '-1']***  
Faça a concatenação dos códigos numéricos conforme o exemplo acima e ao final imprima qual a palavra formada. Após fazer a aplicação, pesquise por códigos ASCII 😉

1. Extra: A Cifra de César foi uma das primeiras técnicas de criptografia criadas pela humanidade. Essa técnica consiste em aplicar um deslocamento de caracteres em uma mensagem, por exemplo, se aplicarmos o deslocamento de 3 caracteres:
   * A letra ‘A’ fica como letra ‘D’;
   * A letra “I” fica como letra “L”;
   * A letra “Z” fica como letra “C”
   * E assim por diante.

A palavra “OI” aplicando o deslocamento de 3 caracteres em cada um dos caracteres ficaria como “RL”.  
Sua tarefa é criar uma aplicação que possa descriptografar a seguinte frase (não aplicamos deslocamento para o caractere espaço): “**HVWRX HPSROJDGR FRP R FXUVR GH SBWKRQ**”. Considere que o deslocamento de caracteres usado foi 3 caracteres.