Hype
Ciência de Dados

Dados Gerais do Processo Seletivo

ABERTURA DO CASE	18/04/2023
DATA LIMITE DE ENTREGA	09/05/2023 até 23h55
CONTATOS	Grupo no Telegram ou e-mail (hype-each@usp.br) - recomendamos utilizar o Telegram

CASE DE PROGRAMAÇÃO

Introdução

Olá candidato(a)! Seja muito bem-vindo(a) ao 1º Processo Seletivo (PS) do Hype de 2023! Nesta primeira fase do PS, você deverá resolver um case técnico. Você recebeu o case de acordo com seu nível de habilidade autodeclarado no formulário de inscrição, que foi o Case de Programação.

É importante que você saiba que envios incompletos também serão considerados. **A nossa maior prioridade com este case é avaliar o esforço e a evolução que você teve.** Sendo assim, não será avaliado somente se suas respostas estão corretas. E não se preocupe, pois fizemos este case tanto para pessoas que não sabem programar quanto para quem já sabe programar. Nós separamos um material de estudo para que você possa consumi-lo para adquirir as habilidades necessárias para resolver o case.

Este case é constituído por exercícios de lógica de programação em Python. Como mencionamos anteriormente, não será cobrado somente se sua resposta está certa ou errada. Sendo assim, você deve deixar comentários no código para descrever o raciocínio lógico por trás dele, para que consigamos entender seu raciocínio. Caso você venha a cometer algum erro, vamos procurar entender o que você estava tentando fazer no código a partir dos seus comentários, e levá-los em consideração na avaliação, apesar

do programa possuir um erro. Caso você não saiba o que são comentários no código, assistir o material de estudo vai fazer com que você aprenda isso!

Importante: incentivamos que você pesquise sobre as dúvidas que você pode ter durante o case em sites, vídeos, guias, outros materiais de estudo, ou quaisquer recursos didáticos na Internet, pois isso pode te ajudar. É possível aprender muito pesquisando na Internet!

Entretanto, saiba que utilizaremos um identificador de plágio, então evite copiar diretamente códigos da Internet ou copiar de outros colegas que estão participando do processo seletivo. A cópia de resoluções inteiramente prontas será vista como plágio e pode levar à reprovação no processo seletivo. Por isso, lembre-se de comentar seus códigos com suas palavras!

Além disso, **caso tenha qualquer dúvida sobre os exercícios ou sobre o PS, você pode perguntar para nós** através do nosso grupo no Telegram chamado "Hype - PS 2023" ou chamar os monitores desse grupo no privado (os monitores são os Diretores do Hype). Você também pode tirar dúvidas através do nosso e-mail (hype-each@usp.br) - porém é preferível que envie dúvidas no Telegram, por ser mais rápido de respondermos. Ver que você está se esforçando e tirando suas dúvidas vai nos mostrar seu interesse em entrar no Hype!

Estrutura e Orientações

O case possui 6 exercícios de lógica de programação em Python, divididos em:

- Nível Iniciante
  - Exercício 0
  - Exercício 1
  - Exercício 2
- Nível Médio
  - Exercício 3
  - Exercício 4
- Nível Difícil
  - Exercício 5

É obrigatório utilizar os nomes das funções iguais aos nomes fornecidos nos enunciados. Além disso, não utilize acentos nos nomes das variáveis e nem nos nomes das funções.

Quanto mais bem escrito o seu código estiver e quanto mais comentários você escrever nele, mais fácil será nosso entendimento da lógica por trás dele. Por isso, dê nomes que façam sentido para suas variáveis e priorize lógicas simples. Códigos ilegíveis não comentados vão afetar seu desempenho.

Indicação de Materiais de Estudo

Este case foi pensado para quem não sabe programação em Python e para quem já sabe. Caso você não saiba programação em Python, vamos recomendar uma playlist no YouTube que criamos para que você possa aprender todos os conhecimentos necessários para resolver este case. **Recomendamos muito que você assista a playlist inteira, pois, além de você aprender com os vídeos, eles contêm dicas de como resolver os exercícios do case. Além disso, caso você seja aprovado no nosso processo seletivo, vamos pressupor que você já tenha entrado em contato com os conhecimentos presentes na playlist.**

Link da Playlist

Formato de Submissão

Quando terminar seu case, envie-o neste forms. Só aceitaremos uma única submissão. Caso você tenha alguma dúvida de como enviar sua resposta, não hesite em nos contatar pelo grupo do Telegram ou email (hype-each@usp.br).

As respostas dos exercícios de lógica de programação em Python devem ser entregues em arquivos Python (.py). Note que esperamos um arquivo por exercício, com a resposta do exercício devidamente identificada no nome do arquivo.

Recomendamos que você faça seus códigos através do Google Colab, pois essa é uma plataforma do Google que permite criar e executar códigos em Python online, ou seja, sem precisar instalar nada no computador! Além disso, o Colab permite que você baixe seu código se você clicar em *Arquivo → Fazer download → Fazer o download de .py*, e o arquivo proveniente desse download já está no formato correto para submissão! Porém, saiba que cada arquivo deve conter a resolução de somente 1 exercício. Os nomes dos arquivos devem identificar as questões que eles resolvem, **conforme a estrutura abaixo** (não precisa enviar os arquivos dos exercícios que você não conseguiu resolver).

Após salvar os arquivos dos exercícios, reúna-os em uma pasta e compacte-a para o formato .zip. A submissão (entrega) deve ser feita por meio de um único arquivo compactado zip com o nome “nusp_nome_sobrenome.zip”. Outros arquivos de compactação como .rar, .cab, entre outros, não serão aceitos. Exemplo de **estrutura** do arquivo zip:

12345678_Felipe_Silva.zip

├ ex0.py

├ ex1.py

├ ex2.py

├ ex3.py

├ ex4.py

├ ex5.py

Lembre-se que envios de códigos incompletos também serão considerados!

Importante: Recomendamos começar pelo Nível Iniciante, pois ele contém as questões mais fáceis. Não se preocupe caso não consiga resolver a questão do Nível Difícil, mas caso você já saiba programar deve resolvê-la. Não fique assustado com o tamanho deste PDF e com o tamanho dos enunciados, pois a maioria dos conteúdos foram colocados justamente para te ajudar na realização dos exercícios! Por fim, ao lado do título de cada exercício é informado quais conhecimentos são necessários para desenvolver uma possível solução para aquele exercício (a nomenclatura é baseada na playlist do material de estudo). Por fim, deixamos exemplos de teste nos exercícios, mas **é importante que você crie também outros testes para ter certeza que seu código está funcionando corretamente.**

NÍVEL INICIANTE

Exercício 0 - Operações Matemáticas

Você acabou de entrar na faculdade e está cursando neste semestre uma disciplina chamada Análise de Dados. Porém, essa disciplina é difícil e você não foi muito bem na primeira prova e nem na segunda prova. Você pede para o professor da disciplina, Luciano, solicitar um trabalho que valha nota, para que você consiga aumentar sua nota. O professor pensa sobre seu caso e faz uma proposta. Luciano te informa que ele ministra outra disciplina chamada Mineração de Dados para outra turma. Na disciplina de Mineração de Dados, os alunos devem fazer 4 trabalhos (T1, T2, T3 e T4). A média final na disciplina de Mineração de Dados é calculada a partir da média ponderada das notas (n1, n2, n3 e n4) que o aluno conseguiu nos trabalhos T1, T2, T3 e T4. A nota do trabalho 1 (identificada por n1) possui peso 1, a nota do trabalho 2 (identificada por n2) também possui peso 1, a nota do trabalho 3 (identificada por n3) possui peso 3, e a nota do trabalho 4 (identificada por n4) possui peso 5. Sendo assim, Luciano pede para que você crie um programa que calcule a média final de um aluno da disciplina de Mineração de Dados. Caso você consiga, ele te dará nota extra.

Escreva a função **media_ponderada(*n1*, *n2*, *n3*, *n4*)**, que recebe as notas *n1*, *n2*, *n3* e *n4* de um aluno e **retorna** o valor de sua média final (lembrando de se tratar de uma média ponderada).

Recomendamos que você pesquise como calcular a média ponderada, caso não se lembre, antes de fazer este exercício.

Lembre-se:

- n1 é a nota do trabalho 1 (T1); n2 é a nota do trabalho 2 (T2); n3 é a nota do trabalho 3 (T3); n4 é a nota do trabalho 4 (T4)
- n1 possui peso 1; n2 possui peso 1; n3 possui peso 3; n4 possui peso 5
- não é necessário arredondar a nota média

Exemplos de testes com entrada e saída:

```
Entrada:
media_ponderada(3,8,4,10)

Saída:
7.3
```

```
Entrada:
media_ponderada(2,7,1,6)

Saída:
4.2
```

```
Entrada:
media_ponderada(7,6,8,9)

Saída:
8.2
```

Exercício 1 - Operações Matemáticas e Condicionais

Você sempre gostou de Geometria na escola. Recentemente, você aprendeu a programar. Sendo assim, você decide criar um programa que recebe o comprimento, em metros, da base e da altura de um triângulo e calcula sua área. A partir da área calculada, você deseja saber se a área do triângulo é maior do que 10 metros quadrados.

Escreva a função **area_tri(*base, altura*)**, em que *base* é a base do triângulo e *altura* é sua altura, ambas em metros. A função deve **retornar** 'maior' caso a área do triângulo seja maior do que 10, ou **retornar** 'menor ou igual' caso não seja maior que 10.

Exemplos de testes com entrada e saída:

```
Entrada:
area_tri(5,2)

Saída:
'menor ou igual'
```

```
Entrada:
area_tri(7,8)

Saída:
'maior'
```

```
Entrada:
area_tri(5,4)

Saída:
'menor ou igual'
```

Exercício 2 - Listas, Repetição "for" e Condicionais

Os alunos da Universidade de São Paulo (USP) são identificados pelo seu Número USP (NUSP). Além disso, cada aluno possui um cartão físico da USP que possui

informações de seu nome, NUSP e seu curso. Infelizmente, é comum que os alunos percam seus cartões enquanto andam pela faculdade ou participam de festas. Quando os cartões perdidos são encontrados por outras pessoas, eles são levados para os “achados e perdidos” da faculdade. Os funcionários dos “achados e perdidos” informam, em um site da USP, o número dos cartões perdidos que estão nos “achados e perdidos”. Você, sortudo como é, perdeu seu cartão USP na terceira semana de aula. Sendo assim, decide entrar no site para ver se seu cartão está nos achados e perdidos. Quando você abre o site, se depara com muitos Números USP de diferentes de pessoas que também perderam seus cartões, de forma que fica muito difícil encontrar seu NUSP no meio de todos aqueles Números USP que estão no site. Você percebe que não somente você possui azar. Há uma lista muito grande de diferentes NUSPs. Você fica com preguiça de olhar um NUSP de cada vez e, por isso, decide criar um programa que diz 'sim' caso seu NUSP esteja no meio daquela lista, e ‘nao’ caso ele não esteja.

Sua tarefa neste exercício é criar a função **encontra_nusp(*seu_nusp*, *lista_nusp*)** que recebe o número inteiro *seu_nusp* que corresponde ao Número USP que será procurado e também a lista *lista_nusp* que contém todos os diversos Números USP que estão no site de “achados e perdidos” (representados por números inteiros). A função deve **retornar** ‘sim’ caso seu NUSP esteja na lista, e **retornar** ‘nao’ caso não esteja.

A seguir segue a lista *lista_nusp* de Números USP que estão no site. Você pode copiar a lista e colar no seu programa para testá-lo.

```
lista_nusp = [91235, 46782, 12345, 42149, 794, 18362, 82016, 11, 5932, 1818, 91375, 246, 1920, 1820, 29, 4823, 2540, 18, 28342, 81230, 92304, 9213, 289, 23819, 98233]
```

Exemplos de testes com entrada e saída:

```
Entrada:
encontra_nusp(12345, lista_nusp)

Saída:
'sim'
```

```
Entrada:
encontra_nusp(28, lista_nusp)

Saída:
'nao'
```

```
Entrada:
encontra_nusp(82016, lista_nusp)

Saída:
'sim'
```

NÍVEL MÉDIO

Exercício 3 - Listas, Repetição "for" e Condicionais

Você é um cientista de dados estagiário em um banco. Seu chefe lhe deu um problema para resolver. Ele te forneceu uma lista com os valores dos lucros mensais do banco desde sua fundação. Seu chefe te deu a tarefa de descobrir o menor lucro desde a fundação do banco. Se esforce para não perder o seu emprego!

Escreva a função **menor_lucro(*lista_lucros*)** que recebe a lista *lista_lucros* que contém todos os valores dos lucros mensais do banco desde sua fundação. Os lucros são representados por números inteiros. Você deve **retornar** o menor lucro do banco.

É **proibido** utilizar o método min(***lista_lucros***), que retorna o elemento da lista com o valor mínimo.

Importante: você deve encontrar o menor valor sem copiar a lista e sem realizar alterações nela. Então, você não poderá utilizar algoritmos de ordenação ou copiá-la para outra lista.

Exemplos de testes com entrada e saída:

```
Entrada:
menor_lucro([5000, 4500, 9700, 10920, 3200])

Saída:
3200
```

```
Entrada:
menor_lucro([7500, 1300, 37000, 2200, 3100, 5000, 10500, 30900])

Saída:
1300
```

```
Entrada:
menor_lucro([10300, 15000, 11400, 14900])

Saída:
10300
```

Exercício 4 - Repetição "for" ou "while"

Você está começando a gostar de programação e deseja fazer algo diferente. Você decide que vai criar um programa que soma os resultados da tabuada de algum número. Os resultados da tabuada do 6 são 6, 12, 18, 24, 30, 36, 42, 48, 54 e 60. Sendo assim, a soma dos resultados da tabuada do 6 é dada por 6 + 12 + 18 + 24 + 30 + 36 + 42 + 48 + 54 + 60 = 330. Você decide criar uma função que automatiza esses cálculos.

Escreva a função **soma_tabuada(*n*)** que recebe o número inteiro *n*. O programa deve somar os resultados da tabuada de *n*, indo de *n * 1* (n vezes 1) até *n * 10* (n vezes 10), e **retornar** essa soma.

Dica: comece fazendo uma função que imprime os resultados da tabuada, e depois modifique a função para somar esses resultados, e retornar a soma.

Exemplos de testes com entrada e saída:

```
Entrada:
soma_tabuada(8)

Saída:
440
```

```
Entrada:
soma_tabuada(10)

Saída:
550
```

```
Entrada:
soma_tabuada(2)

Saída:
110
```

NÍVEL DIFÍCIL

Exercício 5 - Repetição "while" e Divisões (Divisão Inteira e Operador Módulo)

Você tomou gosto pela programação e foi procurar exercícios difíceis em sites com desafios de programação. Eis que surge o problema deste exercício. Dado um número inteiro, você precisa determinar qual é a soma dos dígitos desse número. Por exemplo, a soma dos dígitos do número 1234 é 1+2+3+4=10.

Escreva a função **soma_digitos(*n*)**, em que *n* é um número inteiro. A função deve **retornar** o resultado da soma dos dígitos do número *n*.

Exemplos de testes com entrada e saída:

```
Entrada:
soma_digitos(11)

Saída:
2
```

```
Entrada:
soma_digitos(19234)

Saída:
19
```

```
Entrada:
soma_digitos(645)

Saída:
15
```