Profa. Nádia Félix
Projeto TJ: Python - Análise de Dados
February 10, 2025

Tarefa 1: Tipos de Dados, Funções e Condicionais

Instruções sobre como redigir e enviar seu trabalho estão disponíveis na página do curso. O não cumprimento dessas instruções resultará na perda de pontos. Direcione quaisquer perguntas ao professor.

Leia isto primeiro.

1) Comece cedo! Se você encontrar dificuldades ou tiver dúvidas, é melhor identificar esses problemas com antecedência, e não nas horas anteriores ao prazo de entrega!
2) Caso tenha dúvidas ou encontre problemas, não envie e-mails diretamente ao professor. Publique no fórum de discussão para que seus colegas também possam se beneficiar caso tenham a mesma dúvida.
3) Certifique-se de fazer backup do seu trabalho! Recomenda-se, no mínimo, salvar em uma pasta do Dropbox ou, melhor ainda, usar o Git, que vale o esforço de aprender.

Questão 1. Aquecimento: Definindo Funções Simples

Neste problema, você irá praticar a definição de funções simples em Python.

1) Defina uma função chamada *bird_pad*, que recebe uma string como único argumento e imprime essa string com o prefixo e o sufixo "bird".
   - Exemplo:
     - bird_pad('goat') deve produzir a saída: "birdgoatbird"
     - *bird_pad*('_') deve produzir: "*bird_bird*"
2) Defina uma função chamada *print_n*, que recebe dois argumentos: uma string s e um número inteiro não negativo n (nessa ordem). A função imprime a string s repetida n vezes, cada uma em uma linha separada.
   - Exemplo:
     - *print_n*('*cat*', 3) deve produzir a saída: cat cat cat
     - Dica: Cuidado com o caso onde n é igual a 0.

Questão 2. Algoritmo de Euclides O algoritmo de Euclides é um método para encontrar o maior divisor comum (MDC) de dois números. O MDC de dois números m e n é o maior número que divide ambos.

1) Use o pseudocódigo do algoritmo de Euclides (https://en.wikipedia.org/wiki/Euclidean_algorithm) para implementar uma função chamada gcd, que recebe dois inteiros como argumentos e retorna o MDC deles.
   - Não é necessário realizar verificações de erro nesta função.
   - Nota: Confundir print com return é um erro comum. Preste atenção!
2) Use sua função para calcular o MDC dos seguintes pares de números:
   (a) 1200, 300
   (b) 5040, 60
   (c) 29, 31
   (d) 2023, 2024
3) O que acontece se um ou ambos os argumentos forem negativos? Esse comportamento faz sentido? Execute gcd com números negativos e escreva uma ou duas frases discutindo o que observou.

Questão 3. Aproximando o Número de Euler *e*

O número de Euler ***e*** é definido como a soma infinita:

$e = \sum_{k=0}^{\infty} \frac{1}{k!} = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + ...$

onde k! $= \begin{cases} 1, & \text{se } k = 0 \\ k \cdot (k-1)!, & \text{se } k > 0 \end{cases}$

1) Use a definição de Bernoulli para ***e***:
   - $e = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x$
     Implemente a função *euler_limit*, que recebe um inteiro positivo n e retorna uma aproximação de e tomando $x = n$.
2) Implemente a função factorial, que calcula o fatorial de um número inteiro não negativo k usando recursão. Não use math.factorial, mas pode usá-lo para testar sua implementação.
3) Defina a função $euler\_infinite\_sum$, que recebe um inteiro n e retorna uma aproximação de e somando os primeiros n termos da série infinita acima.
4) Crie a função euler_approx, que recebe um número positivo t e retorna o menor número de termos necessários na série infinita para que a aproximação de e esteja dentro de t do valor real.
5) Defina as funções *print_euler_sum_table* e *print_euler_lim_table*, que imprimem os valores sucessivos das aproximações com base nas funções $euler\_infinite\_sum$ e *euler_limit*, respectivamente.
6) Compare as duas aproximações. Qual delas converge mais rápido para ***e***

Questão 4. Testando Propriedades de um Inteiro Um número inteiro n é uma potência de 2 se $n = 2^p$, onde p é um número inteiro.

- Implemente a função $is\_power\_of\_2$, que recebe um número inteiro e retorna **True** se for uma potência de 2, e **False** caso contrário. Dica: Uma solução elegante pode ser feita com recursão, mas não é obrigatório.
- Generalize a solução anterior com a função $is\_power(b, n)$, que verifica se n é uma potência de b. Por exemplo, $n = b^p$