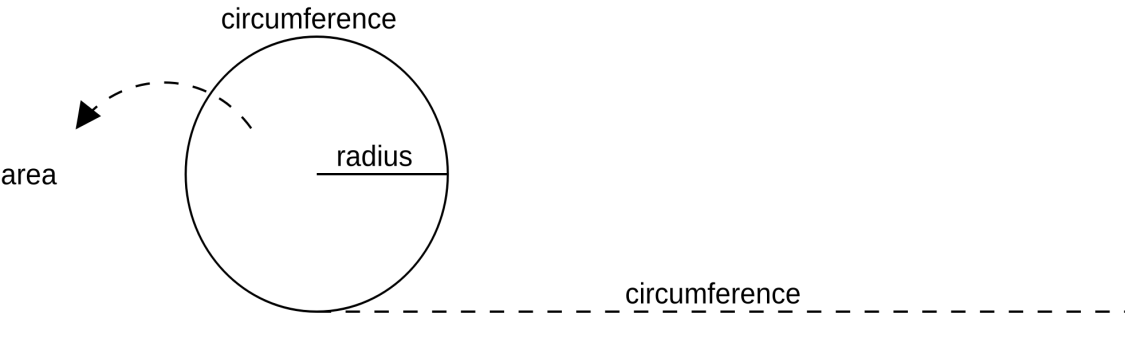
# Lista 2 - Funções e lógica/booleanos

## Submissão: Crie um programa para cada questão abaixo, ou seja um executável como mostrado em aula, depois compacte eles em um arquivo .zip(só aceitarei essa forma de compactação) e submeta ao sigaa.

## Exemplo: Aluno de matrícula 000000 deve criar os arquivos: 000000\_q1.py, 000000\_q2.py, 000000\_q3.py e 000000\_q4.py. Deve compactar usando a técnica zip e submeter ao sistema.

1. **(0,05) PI - Circunferência e área.**

Dado o raio de um círculo, calcule sua circunferência e área.



A circunferência de um círculo com raio r é dada por 2 × π × r ou duas vezes pi vezes o raio.

A área de um círculo com raio r é dada por π × r ² ou pi vezes o quadrado do raio.

O número π, ou pi, é uma constante matemática com um valor irracional de 3,141 592 653 ...

**Entrada e saída**

O seu programa deve [ler](https://panda.ime.usp.br/pythonds/static/pythonds_pt/01-Introducao/09-entradaSaida.html) um valor de raio r, fornecido pelo usuário e número decimal, tal que

0 < r <10000,0 .

Para cada valor fornecido como entrada o seu programa deve produzir dois números, **c** e **a**, indicando circunferência e área. Eles devem ser arredondados para duas casas decimais.

|  |  |
| --- | --- |
| Entradas de exemplo | Saídas de exemplo |
| 1 | 6.28 3.14 |
| 12 | 75.40 452.39 |
| 6.6 | 41.47 136.85 |
| 0.159 | 1.00 0.08 |

**Funções circunferência e área**

Seu programa deve implementar duas funções: circunferencia e area. Cada uma deve receber um número de ponto flutuante e retornar um número de ponto flutuante.

**Observações:**

Mantenha a leitura e saída nos formatos indicado.

Teste com entradas do usuário no formato inteiro e decimal.

Teste usando casos extremos, ou seja, que são válidos mas incomuns.

Para o valor de pi, pode usar math.pi.

1. **(0,05) Soma dos quadrados.**

Suponha que um usuário receba três valores e deseja saber qual a soma dos quadrados dos dois maiores.

O seu programa deve fornecer a ele essa possibilidade.

**Entrada e saída**

O programa deverá ler 3 números naturais, entre 0 e 9, fornecidos pelo usuário, um em cada linha, e fornecer a soma dos quadrados dos dois maiores.

|  |  |
| --- | --- |
| Entrada de exemplo | Saída de exemplo |
| 2  6  3 | 45 |
| 0  0  1 | 1 |
| 8  2  1 | 68 |

**Função soma\_dos\_quadrados**

Seu programa deve implementar uma função soma\_dos\_quadrados que recebe três números naturais como parâmetro e retorna a soma dos quadrados dos dois maiores.

**Observações:**

Não use condicionais. Sua resolução deve ser feita com o conteúdo visto até a aula de lógica e booleanos.

Teste usando casos extremos, ou seja, que são válidos mas incomuns.

1. **(0,075) Ordenação**

Crie um programa que ordena três números naturais x, y e z em ordem crescente. Assim, x terá o menor valor, y o intermediário e z o maior.

**Entrada e saída**

O seu programa deve ler três números naturais, um por linha, e imprimir como saída o valores do menor para o maior, na mesma linha.

|  |  |
| --- | --- |
| Entradas de exemplo | Saída de exemplo |
| 0  45  50 | 0 45 50 |
| 120  400  210 | 120 210 400 |
| 370  200  10 | 10 200 370 |

**Função ordena**

Crie no seu programa uma função ordena que receberá os três valores informados ao usuário, e que imprime a saída no formato desejado, não retornando algo para a main que a chamou.

**Observações**

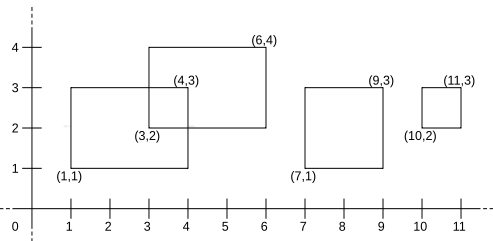
Teste usando casos extremos, ou seja, que são válidos mas incomuns.

Use variáveis temporárias, ou seja, que armazenam temporariamente valores.

Use funções que vimos em aula, até lógica e booleanos. Funções de ordenação embutidas, condicionais e outras coisas mais avançadas são proíbidas.

1. **(0,075) Dois retângulos se cruzam?**

Retângulos ortogonais são aqueles com arestas paralelas aos eixos do plano cartesiano. Dois retângulos podem se cruzar quando há pontos que estão acima ou dentro de ambos. Em outras palavras, eles se cruzam quando compartilham uma superfície comum ou área de fronteira. Isso inclui quando eles estão um dentro do outro. Eles não se cruzam quando não têm pontos comuns.



A figura acima mostra quatro retângulos ortogonais com arestas paralelas aos eixos x e y. Aquele com o canto inferior esquerdo em (1,1) e o canto superior direito em (4,3) cruza com aquele com o canto inferior esquerdo em (3,2) e o canto superior direito em (6, 4). Aquele com o canto inferior esquerdo em (7,1) e o canto superior direito em (9,3) não se cruza com aquele com o canto inferior esquerdo em (10,2) e o canto superior direito em ( 11,3).

Escreva um programa que, dados os cantos inferior esquerdo e superior direito de dois retângulos, calcule se eles se cruzam.

**Entrada e Saída**

A entrada consistirá em várias linhas, cada uma com um inteiro, de x₀, y₀, x₁, y₁, x₂, y₂, x₃ a y₃, onde o primeiro retângulo tem a coordenada inferior esquerda de (x₀, y₀) e a coordenada superior direita de (x₁, y₁) e o segundo retângulo tem a coordenada inferior esquerda de (x₂, y₂) e a coordenada superior direita de (x₃, y₃).

A saída deve ser True ou False, se os triangulos se cruzam ou não, respectivamente.

|  |  |
| --- | --- |
| Exemplo de entrada | Exemplo de saída |
| 1  1  4  3  3  2  6  4 | True |
| 10  2  11  3  7  1  9  3 | False |

**Função cruza**

Crie no seu programa uma função cruza que receberá os valores informados pelo usuário, e que imprime a saída no formato desejado, não retornando algo para a main que a chamou.

**Observações**

Teste os casos extremos.

Tente desenhar alguns valores em papel para criar testes.

Não use condicionais. Sua resolução deve ser feita com o conteúdo visto até a aula de lógica e booleanos.