CIÊNCIA DA COMPUTAÇÃO
UFRJ
BY NC ND

Curso de Computação 1
Introdução à Programação em Python

AULA 2

Na aula anterior aprendemos a definir e utilizar funções em Python. No Python, as funções são a estrutura base de organização de código de um dos estilos de programação mais simples e utilizados em nossos dias, que é a ***programação modular***. Este estilo de programação é uma evolução surgida naturalmente a partir do paradigma de programação ***imperativo estruturado***. Vamos entender um pouco cada um destes conceitos.

1. Paradigmas e estilos de programação

O paradigma de programação determina o padrão para estruturação e execução do programa. Ele dita a visão que o programador deve adotar ao planejar a solução algorítmica para o problema sendo abordado, de forma que possa ser mais facilmente codificado na linguagem de programação. Assim sendo, o paradigma de programação a ser seguido deve ser escolhido antes da escrita do código. Uma curiosidade: algumas linguagens de programação permitem mais de um paradigma.

Neste curso será adotado o paradigma de programação ***imperativo estruturado***, e o estilo ***modular***. O nome "Imperativo" é uma referência ao tempo verbal imperativo, onde o programador diz ao computador, na forma de instruções ou comandos, o que deve ser feito seguindo uma ordem sequencial (faça isso, depois isso, depois aquilo...). O termo "estruturado" refere-se ao uso de estruturas pré definidas para que o controle de execução (ou fluxo de execução) das instruções possa ser um pouco mais sofisticado do que a simples ordem sequencial. Por exemplo, o programador pode lançar mão de uma estrutura específica para que um conjunto de instruções seja repetido (veremos isso mais adiante no curso ;-).

Já o estilo se refere à maneira adotada para organizar o código, e influencia também a linha de raciocínio a ser adotada na expressão algorítmica que será construída para o problema, e posteriormente codificada na linguagem de programação desejada. No estilo modular, a construção de soluções (e consequentemente a organização de código) é norteada pela construção de módulos versáteis de código, ou seja, ao invés de um código grande pra fazer várias coisas, são feitos pequenos códigos (chamados módulos) que se destinam a fazer uma única coisa. A combinação destes módulos é que dá conta de realizar uma grande tarefa. Esse estilo é muito popular, por ter várias vantagens:

- a mais valorizada dentre as vantagens é a *reutilização de código*. Isso porque os módulos podem ser reutilizados na composição da solução de mais de um problema, se combinados de maneiras diferentes.

- outra vantagem importante é a possibilidade de testar cada módulo individualmente, o que é mais eficiente do que testar um código extenso só depois que ele estiver completamente escrito.
- em relação à elaboração de uma solução algorítmica (ou seja, na etapa anterior à construção do código) temos a vantagem de ser uma estratégia que ajuda a nortear o raciocínio.

Na linguagem Python, os módulos são as **funções**. Por isso estamos focando nosso estudo na construção de funções em Python.

2. Uso articulado de funções

As funções são estruturas úteis não apenas para manter o código organizado, mas principalmente, para auxiliar o programador no processo de construção de soluções de programação para os problemas propostos. Usando funções, é possível atacar um problema complexo focando em partes menores deste problema, e articulando as soluções escritas para as partes menores para obter a solução do problema complexo. Veremos a seguir como escrever e utilizar uma ou mais funções para a solução de problemas simples.

Uma breve revisão do que vimos na aula passada: em programação funções são trechos de códigos com o objetivo de produzir um resultado. Assim como na matemática quando você deseja calcular uma função $f(x) = x^2$, na verdade você está realizando uma operação com o objetivo de saber qual é o valor do quadrado de x, uma vez que seja fornecido o valor de x. Em programação funções recebem parâmetros, podem realizar algum cálculo ou produzirem algum processamento e então retornam o valor do resultado.

Na aula passada, construímos algumas funções para realizar operações matemáticas simples (tais como soma, sucessor, quadrado). Essas funções poderiam ser usadas, por exemplo, para a construção de um aplicativo de calculadora. No momento ainda não vamos trabalhar elementos de interface gráfica, mas você já fica sabendo que por trás de um aplicativo de calculadora com interface gráfica, estão trechos de código não muito diferentes dos que já sabemos construir.

Vamos então ver como podemos articular o uso de funções para produzir soluções mais sofisticadas. Considere que deseja-se calcular a área da coroa circular (anel) formada por dois círculos de raio r1 e r2, respectivamente, nos quais r1 > r2 e Pi = 3.14. Na geometria, coroa circular (ou anel) é uma região limitada por dois círculos concêntricos. Se denotamos por r1 o raio da circunferência externa e por r2 o raio da circunferência interna, a área da coroa é dada pela diferença entre a área do círculo externo e a área do círculo interno.

Vamos tentar escrever código para o problema do cálculo da coroa circular. O primeiro que temos a fazer é estudar o problema e planejar nossa solução. Já sabemos que, neste curso,

atacaremos os problemas construindo uma função que retorne o resultado desejado. Vamos então planejar como será essa função:

Problema: calcular a área da coroa circular formada por dois círculos concêntricos de raios r1 e r2.

- O que deve ser feito?

Para responder essa pergunta, tenho que descobrir como é calculada a área da coroa circular.
Pesquisando na internet, um dos links encontrados é: Coroa circular . Através da explicação dada neste site, a área da coroa circular é feita calculando a área do círculo maior, a área do círculo menor, e subtraindo uma da outra. Isso nos faz perceber que, para resolver este problema, temos que atacar também outro problema: calcular a área de um círculo.

- Qual o resultado esperado? (o que será retornado pela função)

O valor numérico que representa a área (essa informação também é obtida estudando as informações do link de referência sobre a coroa circular).

- Quais as entradas?

Temos que fazer o levantamento de que informações devem ser fornecidas para que possamos fazer o cálculo desejado e obter o resultado. Iremos precisar dos raios de ambos os círculos, que podemos chamar de r1 e r2.

- Como faço isso no Python?

  $A_{coroa} = A_{círculo\ maior} - A_{círculo\ menor}$

  Considerando que o círculo maior tem raio r1 e o círculo menor tem raio r2:
  $A_{coroa} = (\pi . r1^2) - (\pi . r2^2)$

  Para fazer essa conta no Python, tenho que escolher um valor para π. Não será interessante pedir pro usuário que deseja saber a área da coroa fornecer como entrada o valor de π, uma vez que esse valor é fixo e não tem a ver com a situação específica do usuário. Para este problema, vamos usar uma aproximação de π com apenas duas casas decimais: 3.14

  Assim sendo, teremos: (3.14 * r1**2) - (3.14 * r2**2)

Passamos então para a fase de codificação e vamos escrever nossa função em Python:

```
def coroa(r1,r2):
    '''Funcao que calcula a coroa circular
    formada pelos circulos de raio r1 e r2
    (r1>r2)'''
    return (3.14*r1**2)-(3.14*r2**2)
```

Esta é uma boa solução. Atende ao problema. Mas será que podemos pensar em uma solução mais organizada e mais fácil de ser construída?

No vídeo a seguir, vamos ver como podemos projetar funções utilizando outras funções já implementadas e a importância desse conceito, para que seja possível a escrita de um código mais coeso e fácil de ser testado.

Curso de Computação 1
Introdução à Programação em Python

- Decomposição de Funções

Reparem que, uma vez que sabemos que funções podem ser reutilizadas, podemos fazer o planejamento da solução de um problema já imaginando a construção de mais de uma função. Isso nos permite trabalhar com pequenos problemas, focando em um de cada vez, ao invés de tentar pensar direto na solução de um problema mais complexo.

Exercício: baixe o arquivo areas.py, que contém, além do código da função pi, da área do círculo e da coroa circular, também a função arearetangulo, que serve para calcular a área de um retângulo. Complemente este arquivo com outras duas funções:

1. Usando uma ou mais funções que já estão prontas nesse arquivo, faça uma função para calcular a área da superfície de um cilindro reto.
2. Usando uma ou mais funções que já estão prontas nesse arquivo, faça uma função para calcular a área do quadrado.

IMPORTANTE: Não pule a etapa de planejamento da solução! Acredite, essa etapa agiliza o seu trabalho e evita erros.

Atividade: Agora você já teve sua primeira experiência de reutilização de código :-). Após terminar este exercício, responda a algumas perguntas sobre ele na atividade "reutilizando código". Fique atento ao prazo de entrega dessa atividade!

Dicas para não errar

Algumas informações importantes quando trabalhamos com o uso articulado de funções:

- O Python só entende que uma função existe uma vez que a definição dela já foi feita. Ou seja, você só vai conseguir usar funções que existam (parece óbvio, mas não custa dizer…). Se quiser usar uma função que ainda não existe, você terá que construí-la.
- Uma vez que uma função foi definida, ela pode ser usada quantas vezes desejarmos. Dessa forma, **só uma definição deve existir para cada função**, mesmo que ela seja usada duas ou mais vezes, ou até mesmo que seja usada por várias outras funções diferentes. Se o Python encontra duas definições para a mesma função, ele guarda a última encontrada apenas, descartando a anterior.

- Os **parâmetros das funções são informações com escopo local**, ou seja, são nomes que dizem respeito apenas ao código da respectiva função, mesmo que haja várias funções em um mesmo arquivo. Isso significa que duas funções distintas podem até usar o mesmo nome para seus parâmetros, porém os parâmetros de uma nada terão a ver com os parâmetros da outra.

Atividade: Para fixar seus conhecimentos, faça agora a atividade "jogo dos erros". Fique atento ao prazo de entrega dessa atividade!

3. Argumentos default

Um outro conceito importante em Python é o da utilização de argumentos default, que são valores padrão fixos para um ou mais parâmetros de uma função. Estes são informados na definição da função. O vídeo a seguir vai mostrar como pode ser interessante usar estes argumentos:

- Argumentos Default

Exercício: Considere a função *potencia* vista no vídeo. O que está errado no seguinte trecho de código:

```
def potencia(y=2,x):
        return x**y
```

4. Tipos de dados

O conceito de tipo de dado é muito importante na programação. Sabemos que uma das coisas que mais fazemos quando programamos é lidar com informações, que manipulamos de maneiras diversas para produzir outras informações. O tipo de dado é como uma forma que a linguagem de programação usa para representar as informações de forma que o computador consiga armazená-las e seguir maneiras padronizadas de manipulá-las. Por exemplo, ao lidar com dados que representam números inteiros, as operações aritméticas são feitas de uma maneira um pouco distinta do que são quando lidamos com dados numéricos que tem parte fracionária. Provavelmente você se recorda que teve que aprender, em algum momento de sua vida escolar, como adaptar as operações de divisão e multiplicação para lidar com números que tinham casas decimais, coisa que não era necessária ao lidar com números inteiros.

Dependendo de seu tipo, a forma de armazenamento do dado será diferente na memória, bem como as operações disponíveis para serem realizadas com esses dados. Veremos no próximo vídeo os principais tipos de dados numéricos.

- Tipos numéricos

Para conseguir lidar com qualquer dado, o Python sempre o enquadra em algum tipo de dado que ele conheça. Quando o programador não diz explicitamente qual tipo de dado ele

deseja que seja utilizado, o Python escolhe um automaticamente, de acordo com regras internas de identificação de padrões.

5. Teste de mesa

Até aqui você aprendeu a como definir funções, fazer reuso, usar argumentos default e tipos numéricos. Quando usamos várias funções para resolver um mesmo problema, nosso código já começa a ficar um pouco mais complexo. Um recurso importante para que a gente não se perca é o **teste de mesa**. O teste de mesa é uma prática que todo programador lança mão quando lida com códigos que não são elementares como os da aula 1. Essa prática consiste em pegar um lápis/caneta e um papel, e tentar reproduzir manualmente o funcionamento do seu código. Ou seja, você vai simular o computador.

Para fazer o teste de mesa, olhe para o código da função que você deseja testar. Vamos considerar o seguinte código:

```
def areacirc(r):
    '''calcula a area do círculo de raio r'''
    return 3.14*r**2

def areacoroa(r1,r2):
    '''calcula a area da cora circular dados os raios r1 e r2 de dois
    círculos concentricos, sendo r1 maior que r2
    '''
    return areacirc(r1)-areacirc(r2)
```

Assim como quando você testa sua função usando o interpretador Python, a primeira coisa que você tem que fazer no teste de mesa é escolher os valores com os quais deseja testar sua função. Vou fazer o teste de mesa da função areacoroa com os valores de entrada 3 e 2. Então pegue seu caderno e escreva no alto da página:

areacoroa(3,2)

Agora, temos que olhar para o código da função areacoroa. Qual é a primeira coisa que o Python vai fazer quando se deparar com a chamada da função areacora(3,2) ? A primeira coisa a ser feita é a associação dos valores 3 e 2 aos parâmetros r1 e r2. Assim sendo, anote essa associação no seu caderno:

areacoroa(3,2)

r1 -> 3, r2 -> 2

E agora, o que a Python fará? Ele vai executar a primeira linha do código da função (a documentação não é executada). Olhando o código, temos que a primeira linha depois da documentação é `return areacirc(r1)-areacirc(r2)`

cc BY NC ND

Curso de Computação 1
Introdução à Programação em Python

Já sabemos que o valor atual de r1 é 3, e o valor atual de r2 é 2. Então vamos anotar no nosso caderno:

```
areacoroa(3,2)
    r1 -> 3, r2 -> 2
    return areacirc(3)-areacirc(2)
```

O que o Python fará a seguir? O valor de retorno da função areacoroa(3,2) já está pronto para ser enviado a quem chamou a função? Ainda não está! O python prossegue trabalhando na execução dessa linha de código, e é isso que nós vamos fazer também. Agora, O Python vai fazer a chamada da função areacirc(3). Isso significa que, até termos o valor de retorno de areacirc(3), a execução da função areacoroa é suspensa. Vamos representar isso no caderno, botando areacirc(3) numa coluna ao lado:

areacoroa(3,2) r1 -> 3, r2 -> 2 return areacirc(3)-areacirc(2)	areacirc(3) r->3

Agora, vamos fazer o teste de mesa de areacirc(3). A primeira coisa é fazer a associação do valor de entrada com o parâmetro. A seguir, olhamos para a primeira linha de código da função areacirc:

areacoroa(3,2) r1 -> 3, r2 -> 2 return areacirc(3)-areacirc(2)	areacirc(3) r->3 return 3.14*3**2

O que o Python faria a seguir? Já temos todas as informações necessárias para fazer a conta do resultado de areacirc(3). É isso que será feito:

areacoroa(3,2) r1 -> 3, r2 -> 2 return areacirc(3)-areacirc(2)	areacirc(3) r->3 return 3.14*3**2 28.26

O valor de retorno dessa função está pronto, e o Python retorna este resultado para quem chamou a função. Vamos fazer isso no nosso teste de mesa:

BY NC ND

Curso de Computação 1
Introdução à Programação em Python

areacoroa(3,2) r1 -> 3, r2 -> 2 return 28.26-areacirc(2)	areacirc(3) r->3 return 3.14*3**2 28.26

Agora o Python prossegue a execução da função areacoroa(3,2), chamando a função areacirc(2).

areacoroa(3,2) r1 -> 3, r2 -> 2 return 28.6-areacirc(2)	areacirc(3) r->3 return 3.14*3**2 28.26 areacirc(2) r ->2 return 3.14*2**2 12.56

E finalmente, o Python retorna o resultado da função areacirc(2) e prossegue com a execução de areacoroa(3,2)

areacoroa(3,2) r1 -> 3, r2 -> 2 return 28.6-12.56 15.70	areacirc(3) r->3 return 3.14*3**2 28.26 areacirc(2) r ->2 return 3.14*2**2 12.56

Vemos então, com nosso teste de mesa, que o valor retornado seria 15.70. O teste de mesa serve para compreendermos o que de fato está sendo feito por nosso código, e é particularmente útil quando testamos nosso código no computador e percebemos que resultado não corresponde ao que esperamos, porém não sabemos dizer onde está o erro. Olhando linha após linha, dessa maneira sistematizada e documentada, iremos certamente encontrar o problema :-)

Exercício: Como exercício, faça o ***teste de mesa*** do seguinte código e ***verifique se ele faz efetivamente o que deveria fazer***:

```
def pi():
    '''retorna o valor de pi com precisao de duas casas decimais'''
    return 3.14

def areacirc(r):
    '''calcula a area do círculo de raio r'''
    return pi()*r**2

def areacoroa(R,r):
    '''calcula a area da cora circular dados os raios R e r de dois
    círculos concentricos, sendo R maior que r
    '''
    return areacirc(R-r)
```

Caso tenha dúvidas, consulte o seu professor/monitor durante a aula síncrona, ou nos horários síncronos e plantões de monitoria.

6. Módulos

As funções que construímos podem ser usadas várias vezes. Para que nosso código fique organizado, é comum colocar funções que tratam de uma mesma classe de problemas (por exemplo, funções para cálculos de polinômios, funções geométricas, etc) em um mesmo arquivo. Esse tipo de organização é feito em várias linguagens de programação, e ficou popularmente conhecido como “biblioteca de funções”. No Python, a esse tipo de arquivo que organiza funções acerca de um mesmo tema dá-se o nome de módulo.

No próximo vídeo, vamos aprender o conceito de módulos presentes no Python. No geral, módulos agrupam funções que realizam tarefas comuns, como cálculos matemáticos, geração de números aleatórios, manipulação de dados temporais (que representam datas e horas), dentre outros.

Curso de Computação 1
Introdução à Programação em Python

- Módulos no Python

O próximo vídeo contém mais alguns exemplos de como criar e reutilizar funções Python e também o uso de funções presentes no Módulo Math.

- Uso de módulos

Por fim, o próximo vídeo contém alguns erros comuns que podem ocorrer quando usamos funções e módulos em Python.

- Erros comuns

Pois bem, até aqui você aprendeu mais sobre a escrita e o uso de funções em Python para produzir soluções computacionais para problemas. Aprendeu a planejar soluções para um problema que podem ser compostas de várias funções. Aprendeu sobre como criar e utilizar argumentos default, e sobre os principais tipos de dados numéricos presentes no Python. Você também aprendeu que a linguagem Python permite o uso de bibliotecas de funções chamadas módulos, e que já temos disponível, junto com a instalação do Python vários módulos com funções pré-definidas que podem facilitar a construção de códigos. Por fim, você viu e aprendeu em “mais exemplos”, um resumo de como definir funções fazendo reutilização e aplicando funções do módulo math e também verificou alguns erros comuns.

Atividade: Para refletir sobre estes conteúdos e fixar seus conhecimentos, faça agora a atividade “Pesquisando funções pré definidas”. Não hesite em tirar dúvidas na aula síncrona ou nos plantões de monitoria.

Concluído o estudo dirigido, leve suas dúvidas para a aula síncrona.Você terá oportunidade de consolidar seus conhecimentos realizando a **atividade prática** desta aula, que ficará disponível no painel de atividades.