Curso de Computação 1
Introdução à Programação em Python

AULA 10

Interação com usuário e programa principal

No curso introdutório de computação, focamos no desenvolvimento de códigos para implementar soluções para pequenos problemas computacionais. Até agora, desenvolvemos várias funções individuais para realizar cálculos específicos, mas ainda não sabemos juntar essas funções para desenvolver um programa aplicativo. Para desenvolver um aplicativo bem simples ainda precisamos de duas coisas:

(1) fazer com que o código Python interaja com o usuário (pedir dados e fornecer respostas em formatos que sejam úteis para o usuário); e

(2) Projetar código que articule estas partes menores e coordene as funcionalidades com a interação com o usuário.

Existem várias alternativas tanto para (1) quanto para (2). Nosso objetivo para este curso é conseguir entender os conceitos básicos acerca destes temas e conseguir trabalhar com as opções mais simples que o Python oferece para interface com o usuário (1) e projeto de aplicações (2).

1. Projeto de aplicações

Assim como em situações onde temos que desenvolver um trabalho maior, que pode envolver várias fases e várias pessoas, o desenvolvimento de softwares complexos também é visto como um processo que deve seguir padrões para evitar os riscos de algo dar errado. Apesar de ainda não estarmos desenvolvendo softwares complexos, nosso códigos já estão ficando maiores e portanto é chegado o momento de entender os conceitos básicos de projeto de aplicações.

Para tirar proveito de toda a experiência acumulada nos processos de desenvolvimento de software, a comunidade criou ***padrões arquiteturais***, ou seja, tipos de arquitetura que já foram testadas e aprimoradas e que mostraram funcionar bem dentro de certos contextos. Os padrões arquiteturais propõem uma organização de mais alto nível para sistemas de software, incluindo seus principais módulos e as relações entre eles. Essas relações definem, por exemplo, que um módulo A pode (ou não pode) usar os serviços de um módulo B.

Dentre os padrões mais adotados atualmente, está o ***padrão de arquitetura em camadas***. De forma simplificada, uma arquitetura em camadas particiona a complexidade envolvida no desenvolvimento de um sistema em componentes menores (as camadas), definindo também as dependências entre essas camadas. Isso ajuda no entendimento, manutenção e evolução

de um sistema. Torna-se mais fácil trocar uma camada por outra: por exemplo, trocar a camada de interface web por uma camada de interface para dispositivo móvel, e assim adaptar uma aplicação web rodar como app no celular, reutilizando as demais funções já escritas que implementam as funcionalidades da aplicação.

A maneira como estamos construindo nossos códigos até agora, organizando as funcionalidades em funções de código e bibliotecas (módulos), é adequada para ser utilizada com um projeto de aplicação em camadas. Como ainda estamos fazendo códigos não muito complexos, vamos trabalhar com apenas duas camadas: a camada de serviço (funções que implementam as funcionalidades da aplicação) e a camada de apresentação, que irá coordenar o uso das funções da camada de serviço de forma a produzir e exibir o resultado que o usuário da aplicação espera.

Apesar de serem menos usadas atualmente do que outras opções mais sofisticadas, as opções simples são o primeiro passo para que aprendamos a lidar com estes aspectos do desenvolvimento de software. Além disso, às vezes o mais simples já resolve seu problema.

1.1. Função principal e organização de código

Voltemos então para os pequenos programas que queremos desenvolver nesta disciplina.

Sabemos como criar as diferentes funções que vão realizar cada parte da tarefa, e agora precisamos de uma função que coordene a execução das diversas funções necessárias para a execução da tarefa. Em Python usa-se como convenção uma função especial que tem especificamente este objetivo, chamada **função principal**, ou **main().** Essa função coordenadora é uma espécie de maestro. Sua sintaxe é parecida com a de qualquer outra função, mas o seu papel é diferente. Vejamos um exemplo no vídeo a seguir.

Vídeo: Laboratório de Física

Observação: Quando escrevemos a chamada de uma função qualquer diretamente no módulo (ou seja, no arquivo onde estão as definições de função), **a função chamada é executada quando damos o *Run* do arquivo, mas não tem seu resultado visível no shell**. Você pode experimentar com um arquivo de laboratório qualquer. Altere o arquivo incluindo uma chamada válida para uma das funções do arquivo. Você não verá o retorno da função no shell, mas ela foi executada. Se você alterar a chamada para algo inválido, tal como número de argumentos diferente do número de parâmetros da função, você verá que o interpretador Python realmente tentou executar a função chamada.

Curso de Computação 1
Introdução à Programação em Python

O nome de função “main” tem um papel especial para o Python, assim como para outras linguagens de programação, sendo entendida como a função que será o ponto de partida para a execução do código. Quando um código tem função main, isto significa que o programador pensou em um cenário de uso completo onde o usuário, ao acionar a função main, terá todo um conjunto de funcionalidades para resolver um problema, o que inclui maneiras de escolher as funcionalidades que deseja usar, interface adequada para fornecer os dados de entrada e ver os resultados gerados. É quando o código recebe o nome de **programa**, **aplicação** ou **aplicativo**.

A função *main* possui um comportamento diferente também quanto ao recebimento de argumentos, mas não vamos ver todos os detalhes disso agora. Para todos os efeitos, nos nossos exemplos, usaremos a função main sem parâmetros. Por causa disso, foi necessário definir todos os dados do experimento diretamente na função main. Para mudar qualquer informação, o código precisará ser editado, alterando os valores das variáveis desejadas. Abaixo você pode ver o código da função e o lugar onde os dados foram explicitamente colocados:

```
def main():
    ''' programa principal para realização dos cálculos do laboratório de cinemática'''

    # dados do experimento
    velocidade_inicial = 10.00
    aceleracao = 0.50
    tempos = [0.00, 1.00, 2.00, 5.00, 10.00, 15.00, 30.00]
    velocidades_esperadas = calcula_esperadas(velocidade_inicial, aceleracao, tempos)

    # resultados observados
    velocidades_observadas = [10.00, 11.10, 12.00, 13.05, 14.00, 15.76, 16.00]

    # cálculo dos erros
    erros = erro_ao_logo_do_tempo(velocidades_esperadas, velocidades_observadas, tempos)

    # retorno(?)
    return erros

main()
```

Quando o usuário final não é o próprio programador, fica bem ruim ele ter que abrir o código para fazer alterações diretamente quando deseja informar os valores com os quais o programa vai trabalhar. Temos que pensar que o usuário final de um programa normalmente não quer se preocupar com a sintaxe esperada pela linguagem para cada tipo de dados que ele precisa fornecer para o programa. Por exemplo, se eu preciso fornecer uma lista de números, não preciso saber que uma lista precisa estar entre colchetes, ou se forneço um nome, não preciso colocar o nome entre aspas…

E finalmente, queremos poder apresentar o resultado para o usuário também de maneira mais amigável. O que é melhor: mostrar uma matriz assim, [[1,2,3],[4,5,6]], ou no formato usual de matrizes, com linhas e colunas sendo algo visível e não apenas uma abstração?

Esses requisitos nos levam ao próximo assunto, a necessidade de prover interação com o usuário, mesmo em programas simples como o do laboratório de Física. As funções básicas de interação com o usuário apresentadas a seguir, usadas juntamente com a função main, vão nos permitir desenvolver pequenos programas com interfaces bem simples para nos ajudar a resolver tarefas cotidianas.

2. Interação com o usuário na interface textual do shell

Ao longo do curso interagimos com nossos códigos através de passagem de parâmetros e valor de retorno para as nossas funções, mas essa não é a maneira adequada de interagir com o usuário final. A interação com o usuário é uma parte importante do projeto do software a ser desenvolvido, e passa por várias decisões acerca de questões de público alvo, dispositivo alvo, e usabilidade pretendida do software.

Você já deve ter experimentado alguns tipos distintos de interface ao usar aparatos tecnológicos. Atualmente a maioria dos celulares, computadores e smartTVs são acessados através de interfaces gráficas, embora distintas entre si. Os aplicativos de um celular, por exemplo, tampouco têm todos a mesma interface, mesmo que seja possível perceber elementos comuns à interface de vários aplicativos.

O projeto de interface de um software tem bastante impacto em sua utilização. Há profissionais do ramo design que se especializam nisso. Em grandes projetos, uma equipe envolvendo profissionais de design e de computação trabalham juntos para produzir a interface. O padrão de arquitetura de software em camadas permite que o desenvolvimento das funções de serviço seja feito em paralelo com o desenvolvimento de uma interface, ou até de mais de uma interface, já que muitas vezes um mesmo aplicativo é disponibilizado para dispositivos diferentes.

A primeira interface com a qual vamos lidar é a interface textual do shell do interpretador Python. Apesar de não serem as mais bonitas, interfaces textuais são muito simples de implementar, e não requerem conhecimento de design gráfico. Por sua simplicidade e eficiência, elas ainda são bastante usadas em situações onde não vale a pena fazer um investimento maior em interfaces gráficas.

De forma geral, uma interface textual se comunica com o usuário em um esquema dialógico, de perguntas e respostas. Temos então que mostrar ao usuário as perguntas que desejamos que ele responda e dados que ele precisa conhecer, coletar as respostas, e quando tivermos os resultados, comunicar esses resultados ao usuário. Para isso, a interface textual precisa essencialmente de apenas dois comandos: um para mostrar informações ao usuário (saída de dados), e outro para receber as informações que o usuário digita (**entrada de dados**).

De forma geral, uma interface textual se comunica com o usuário em um esquema dialógico, de perguntas e respostas. Temos então que mostrar ao usuário as perguntas que desejamos que ele responda e dados que ele precisa conhecer, coletar as respostas, e quando tivermos os resultados, comunicar esses resultados ao usuário. Para isso, a interface textual precisa essencialmente de apenas dois comandos: um para mostrar informações ao usuário (saída de dados), outro para receber as informações que o usuário fornece (entrada de dados).

2.1 Saída de dados

Para mostrar informações ao usuário, o Python oferece uma função de escrita na interface textual, chamada **print**. Nos nossos exemplos, essa interface textual é a janela do shell. Veja o vídeo abaixo e entenda como usamos a função **print** para escrever saídas impressas em texto na tela.

Vídeo: Saída de dados: print

Algumas observações importantes sobre o uso do print:

- Por questões de boas práticas de programação, não usar print dentro de funções que retornam valor (efeito colateral).
- Além disso, o comando `return print ('alguma coisa')` não retorna nada, pois retorna o resultado da função `print`, que é `None` (nada). Essa linha de código está errada! Se você quer que a função retorne 'alguma coisa', tem que fazer apenas

```
return 'alguma coisa'
```

É bastante útil saber um pouco sobre formatação de strings para fazer melhor uso do print. O Python oferece algumas maneiras de fazer isso. Vamos ver bem rapidamente o uso da função *str.format* para formatar strings: *str.format(formatString, p0, p1, ...,)* , retorna uma string formatada segundo a formatString, contendo os dados indicados em p0, p1, ...

A maneira mais fácil de entender o uso do *format* é através de exemplos.

Curso de Computação 1
Introdução à Programação em Python

```
>>> str.format('A soma de {0} e {1} eh {2}', 2, 3, 2+3)
'A soma de 2 e 3 eh 5'
```

Entre chaves {} estão os índices das informações que devem ser mostradas. Nesse caso, temos três dessas informações que serão introduzidas na string no momento da impressão:

- 2, que vai ser incluído no lugar das chaves indicando o índice 0;
- 3, que vai ser incluído no lugar das chaves indicando o índice 1; e
- o resultado da operação 2+3, que vai ser incluído no lugar das chaves indicando o índice 2.

Lembrando que o primeiro índice é sempre zero!

```
>>> str.format('A soma de {1} e {2} eh {3}', 2, 3, 2+3)
      Traceback (most recent call last):
  File "<ipython-input-44-287c69f8968c>", line 1, in <module>
    str.format('A soma de {1} e {2} eh {3}', 2, 3, 2+3)
      IndexError: tuple index out of range
```

A ordem em que os parâmetros são fornecidos não precisa ser a mesma na qual eles são usados. O que importa é que p0,p1,... são parâmetros posicionais. O primeiro parâmetro vai ser referenciado pelo índice 0, o segundo, pelo índice 1, e assim por diante.

```
>>> str.format('{2} eh a soma de {0} e {1}', 2, 3, 2+3)
'5 eh a soma de 2 e 3'
```

Podemos omitir os índices dos dados dentro das chaves caso apareçam na mesma ordem nos argumentos.

```
>>> str.format('A soma de {} e {} eh {}', 2, 3, 2+3)
'A soma de 2 e 3 eh 5'
```

Podemos usar códigos específicos para definir o formato dos dados a serem inseridos na string.

```
>>> str.format('A soma de {0:3d} e {1:3d} eh {2:5d}', 200, 37, 200+37)
'A soma de 200 e  37 eh   237'
```

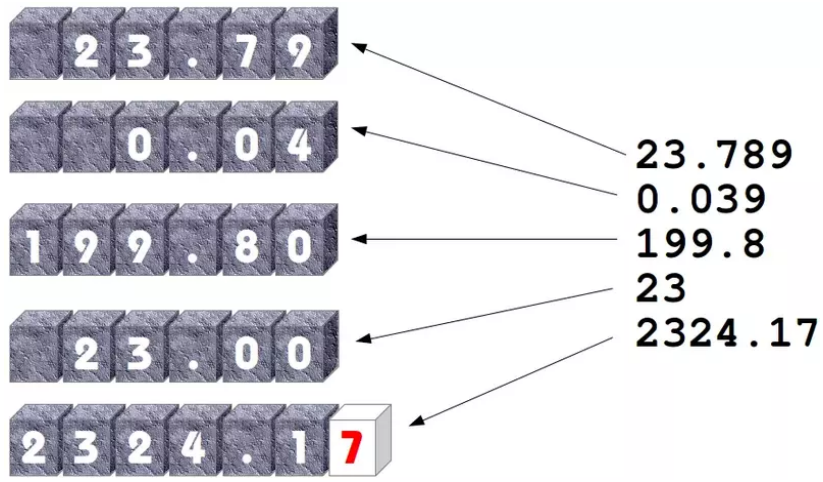
- Formato: {<indice da informação>: <codigo de formatacao> }
- Consulte a documentação da função str.format para conhecer todos os códigos disponíveis.
- Alguns padrões para compor o código de formatação:

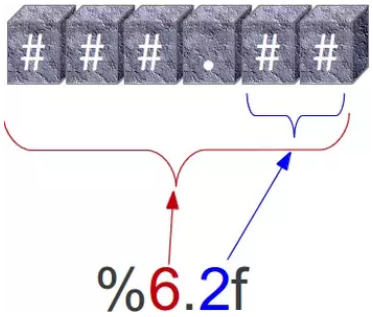
- d para inteiro no formato decimal ou f para float,

- Nd para inteiro decimal com N dígitos
- {N1.N2f} indica dígitos da parte inteira e da parte fracionária

```
>>> str.format('A soma de {0:3.2f} e {1:3.3f} eh {2:3.4f}', 2, 37.005, 2+37.005)
'A soma de 2.00 e 37.005 eh 39.0050'
```

Todos os dados float abaixo foram formatados com o código {6.2f}:





Fonte: https://www.python-course.eu/python3_formatted_output.php

2.2 Entrada de dados

Para a entrada de dados durante a execução, podemos utilizar a função **input**. Veja no vídeo abaixo como utilizar a função **input** e como ela nos possibilita ler dados diferentes, inseridos diretamente pelo usuário, a cada execução de nosso código.

Video: A função input

A função **input** lê dados do usuário e para isto ele temporariamente congela a execução do código, esperando a resposta do usuário até que a tecla Enter seja acionada. É possível passar uma mensagem na forma de uma string, que é impressa na tela, antes que a execução fique em espera. Assim, o usuário fica sabendo que ação o programa está esperando dele para continuar a sua execução.

Atenção: Não use a função *input* dentro de funções que não sejam específicas para entrada de dados. As entradas para uma função devem vir pelos parâmetros. Isso é essencial para que possamos seguir a arquitetura de software em camadas, onde a interface está claramente separada das funções de serviço.

Atividade: Vamos agora treinar um pouco com exercícios de fixação. Responda as perguntas da atividade “Funções print e input”. Fique atento ao prazo de entrega dessa atividade!

2.3 Coordenação da interface com as funções de serviço na função main

Agora já temos todos os elementos que precisamos para a produção de um programa completo! Veja a seguir dois exemplos:

Video: Programinha para gerar uma lista de números aleatórios

Vídeo: Programa para cálculo de soma e subtração de matrizes

Atividade (sem entrega): Faça o download dos programas apresentados nos vídeos anteriores e execute-os no IDLE, procurando visualizar em paralelo o código sendo executado e o comportamento da interface no shell. Procure passar dados válidos e inválidos e verifique o comportamento do programa em cada caso.

Curso de Computação 1
Introdução à Programação em Python

Atividade: Faça o download do programa disponibilizado para o exemplo do "laboratório de física" (o arquivo se chama "laboratório_fisica_interface_simples_a_completar.py"), visto nesta aula. O código foi disponibilizado junto com este roteiro. Seguindo o padrão de arquitetura que separa a interface das funções de serviço, neste programa, as chamadas para as funções de entrada e saída de dados (input e print) estão dentro da função main. Porém faltam algumas! Complete as que estão faltando (indicadas nos comentários do código) de forma a produzir o seguinte efeito quando o usuário utiliza:

```
========= RESTART: /Users/anamaria/Downloads/lab_fisica_simples_b.py =========
Velocidade inicial (m/s): 2
Aceleração constante (m/s²): .5
Tempos (Digite enter quando terminar):
 - Tempo (s): 1
 - Tempo (s): 2
 - Tempo (s): 3
 - Tempo (s):
Velocidades observadas (Espera-se 3 velocidades):
 - Velocidade (m/s) em 1.0 s: 2.2
 - Velocidade (m/s) em 2.0 s: 2.5
 - Velocidade (m/s) em 3.0 s: 3


Comparação de velocidades (v0 = 2.0 m/s; a = 0.5 m/s²):
 - Velocidades esperadas  (m/s): [    2.50,     3.00,     3.50]
 - Velocidades observadas (m/s): [    2.20,     2.50,     3.00]
 - Erros                   (%): [  12.00%,   16.67%,   14.29%]
>>>
```

Prática em Programação

Após concluir as etapas anteriores deste roteiro, faça as atividades práticas desta aula, disponíveis no Google Classroom da turma.
Até a próxima aula!