PROGRAMAÇÃO 1

Aula 4 - Funções

Prof. Emanoel Barreiros

Funções

- Você já utilizou diversas funções até agora (`print()`, `input()`, `len()`, etc.)
- Além de usar funções pré-definidas, você pode criar suas próprias funções
- Funções são trechos de código que podem executar sempre que você desejar

Exemplo 1

```
1    def hello():
2        print('Iaê!')
3        print('Iaê!!!')
4        print('Iaê, brother!')
5        print()
6
7    hello()
8    hello()
9    hello()
```

Funções

- Funções ajudam a encapsular comportamentos recorrentes
- Facilitam manutenção do código
- Reduz ocorrência código repetido
- Se bem compartimentalizado, ajuda a entender melhor o código

Funções com parâmetros

- Ao definir funções você pode definir parâmetros a serem usados pela função
- Lembre das funções no mundo matemático
  - f(x) = 3x + 5

- Ao definir a função, você declara os parâmetros da função
- Ao utilizar a função, os valores passados são chamados de argumentos
- Os argumentos são atribuídos às variáveis definidas como parâmetros

Parâmetros vs argumentos

- Parâmetros nada mais são que variáveis
- Parâmetros tem escopo local, são visíveis apenas dentro das funções onde são definidos
- Argumentos são os valores passados para funções, atribuídos a parâmetros
- Argumentos podem ser valores diretos, expressões ou variáveis

Exemplo 2

```
1    def hello(nome):
2        print('Olá, ' + nome)
3
4    hello('Emanoel')
5    hello('Carlos')
```

Retorno de funções

- Quando a função `len()` é chamada passando-se a string ‘UPE’, ela será avaliada como o valor 3
- Esse valor é o que chamamos de retorno da função
- Funções podem ter retorno ou não (não é obrigatório)
- Utilizamos a instrução `return` para indicar o que deve ser retornado para quem *chamou* a função
- Qualquer expressão pode ser utilizada na instrução `return`
  - Variáveis, expressões propriamente ditas, chamadas a outras funções, em resumo, qualquer coisa que seja avaliada para um valor

Exemplo 3

- Ver código em: https://github.com/emanoelbarreiros/programacao1/blob/master/codigo/aula4/exemplo3.py

O valor None

- Em Python, o valor None representa a ausência de valor
- A maioria das linguagens modernas apresentam uma forma de representar este tipo de valor
  - Em outras linguanges podemos encontrar `nil`, `null`, `undefined`...
- O valor None em Python começa com letras maiúscula
- Funções que não retornam nada na realidade retornam o valor None

Argumentos nomeados

- A maioria dos argumentos é identificada pela sua posição na chamada à função
- A chamada `random.randint(1, 10)` tem resultado diferente de `random.randint(10, 1)`
- Argumentos nomeados, por outro lado, são identificados por seu nome antes do valor passado como argumento
- Argumentos nomeados geralmente são usados para parâmetros opcionais

Exemplo 4

```
1    print('Hello')
2    print('World')
3
4    print('Hello', end='')
5    print('World')
6
7    print('gatos', 'cachorros', 'ratos')
8    print('gatos', 'cachorros', 'ratos', sep=', ')
```

Escopo local e global

- Variáveis declaradas dentro de uma função possuem visibilidade apenas dentro da função onde foram declaradas
- Dizemos que essas variáveis são de escopo local
- Variáveis que existem fora de todas as funções tem visibilidade global
- Dizemos que essas variáveis são de escopo global
- Variáveis só podem ser locais ou globais, nunca ambos
- Escopos são como *containers*, uma vez que a função é encerrada, o container é fechado e todas as variáveis locais são destruídas

Escopo local e global

- Código em um escopo mais externo não consegue visualizar variáveis em um escopo mais interno
- Código em um escopo mais interno consegue visualizar variáveis em um escopo mais externo
- Variáveis diferentes podem ter o mesmo nome se pertencem a escopos diferentes

Escopo local e global

- Variáveis locais não podem ser usadas em um escopo global
  - Exemplo 5: https://github.com/emanoelbarreiros/programacao1/blob/master/codigo/aula4/exemplo5.py
- Escopos locais não podem usar variáveis de outros escopos locais
  - Exemplo 6: https://github.com/emanoelbarreiros/programacao1/blob/master/codigo/aula4/exemplo6.py
- Variáveis globais são visíveis em escopos locais (leitura)
  - Exemplo 7: https://github.com/emanoelbarreiros/programacao1/blob/master/codigo/aula4/exemplo7.py

Variáveis globais e locais com o mesmo nome

- Tecnicamente é possível ter variáveis globais com o mesmo nome de variáveis locais
- Variáveis locais tem prioridade sobre variáveis globais
- Exemplo 8: https://github.com/emanoelbarreiros/programacao1/blob/master/codigo/aula4/exemplo8.py

Instrução `global`

- Caso seja necessário modificar o valor de uma variável global, utiliza a instrução global
  - `global nome_da_variável`