Funções de Programação

Semana 10

Introdução

O que o código abaixo faz?

In [5]:
```
from math import sqrt

n = 5
lst = []
for x in range(1, n+1):
    for y in range(1, n+1):
        lst.append((x, y))

for cat1, cat2 in lst:
    print("{}\t{}\t{}".format(cat1, cat2, sqrt(cat1 ** 2 + cat2 ** 2)))
```

```
1	1	1.4142135623730951
1	2	2.23606797749979
1	3	3.1622776601683795
1	4	4.123105625617661
1	5	5.0990195135927845
2	1	2.23606797749979
2	2	2.8284271247461903
2	3	3.605551275463989
2	4	4.47213595499958
2	5	5.385164807134504
3	1	3.1622776601683795
3	2	3.605551275463989
3	3	4.242640687119285
3	4	5.0
3	5	5.830951894845301
4	1	4.123105625617661
4	2	4.47213595499958
4	3	5.0
4	4	5.656854249492381
4	5	6.4031242374328485
5	1	5.0990195135927845
5	2	5.385164807134504
5	3	5.830951894845301
5	4	6.4031242374328485
5	5	7.0710678118654755
```

Agora analise este código.

In [7]:
```
def obter_lst_de_catetos(n):
    lst = []
    for x in range(1, n+1):
        for y in range(1, n+1):
            lst.append((x, y))
    return lst

def hipotenusa(cat1, cat2):
    return sqrt(cat1 ** 2 + cat2 ** 2)

def imprimir_hipotenusas(lst_catetos):
    for cat1, cat2 in lst_catetos:
        print("{}\t{}\t{}".format(cat1, cat2, hipotenusa(cat1, cat2)))
```

```
lst_catetos = obter_lst_de_catetos(5)
imprimir_hipotenusas(lst_catetos)
```

```
1       1       1.4142135623730951
1       2       2.23606797749979
1       3       3.1622776601683795
1       4       4.123105625617661
1       5       5.0990195135927845
2       1       2.23606797749979
2       2       2.8284271247461903
2       3       3.605551275463989
2       4       4.47213595499958
2       5       5.385164807134504
3       1       3.1622776601683795
3       2       3.605551275463989
3       3       4.242640687119285
3       4       5.0
3       5       5.830951894845301
4       1       4.123105625617661
4       2       4.47213595499958
4       3       5.0
4       4       5.656854249492381
4       5       6.4031242374328485
5       1       5.0990195135927845
5       2       5.385164807134504
5       3       5.830951894845301
5       4       6.4031242374328485
5       5       7.0710678118654755
```

Funções

As **funções** em Python permitem dividir uma tarefa em subtarefas. Na essência, uma função agrupa um conjunto de um ou mais comandos. No entanto, quando esses comandos forem agrupados para resolver uma tarefa específica, as funções permitem que a programação seja feita em um nível de abstração mais alto. As funções permitem a criação de programas mais complexos.

Definindo uma função

A palavra reservada `def` indica que o restante da linha define o **nome da função** e, se houver, seus **parâmetros**. As próximas linhas que estiverem indentadas pertencem à função.

A sintaxe da definição da função é a seguinte.

```
def <nome-da-função>(<parâmetros>):
  <corpo-da-função>
```

In [15]:
```
# Exemplo de função sem parâmetros e sem retorno explícito
def saudar1():
    print("Olá")
    print("Hello")
#print("Hola") # Não faz parte do corpo da função
```

In [49]:
```
# Exemplo de função com um parâmetro, mas sem retorno explícito
def saudar2(nome_da_pessoa):
    print("Olá " + nome_da_pessoa)
saudação = saudar2("Jeff")
print(saudação)
```

```
Olá Jeff
None
```

O comando `return`

As funções podem retornar valores para que a chama. Esse envio do valor para quem a chamou é feito pelo comando `return` .

```
return <expressão>
```

In [55]:
```
def enviar_para_twitter(saudação):
    # enviar para o twitter
    return True
```

In [57]:
```
# Exemplo de função com um parâmetro e com retorno explícito
def saudar3(nome_da_pessoa):
    return "Olá " + nome_da_pessoa
saudação = saudar3("Jeff")
enviou = enviar_para_twitter(saudação)
enviou
```

Out[57]:
```
True
```

Exercício de fixação

- Explique com suas palavras o que é uma função de programação e quais suas vantagens.
- Pesquise e explique com suas palavras o que é um parâmetro de função de programação.
- O que faz o comando `return` em Python.
- Explique o que é o corpo da função.

Chamando uma função

Definir uma função não faz com que ela seja executada. É necessário fazer uma *chamada à função*. No entanto, uma função não pode ser chamada se não for estiver definida. As funções podem ser definidas pelo próprio programador como no caso de `hipotenusa(x, y)` , numa biblioteca externa como no caso de `sqrt(n)` , que fica na biblioteca `math` ou ainda ser fornecida pela própria linguagem de programação como no caso de `print()` , `float()` e `input()` .

A sintaxe da chamada de função é a seguinte.

```
<nome-da-função>(<argumentos>)
```

In [24]:
```
saudar1()
```

```
Olá
Hello
```

In [32]:
```
nome = "Paula"
saudar2("Paula")
```

```
Olá Paula
```

In [31]:
```
print(saudar3("Jeff"))
```

```
Olá Jeff
```

Exercício de fixação

- Pesquise e explique o que é um argumento de função em programação.

Argumentos são os valores passados na chamada de função.

- Por que é necessário incluir um nome (i.e., string) como `"Jeff"` em `saudar2()` e `saudar3()` ?

In [35]:
```
saudar3("Jeff")
```

Out[35]:
```
'Olá Jeff'
```

In [36]:
```
saudar2("Jeff")
```

```
Olá Jeff
```

- Compare atentamente a saída de `saudar2()` e `saudar3()` .

`saudar2()` não possui retorno explícito (i.e., retorna `None`) ao passo que `saudar3()` retorna uma string quando é passada uma string como parâmetro.

- Como a definição da função `saudar2()` deve ser alterada se fosse necessário saudar duas pessoas ao invés de uma única?

In [40]:
```
def saudar2(nome_da_pessoa1, nome_da_pessoa2):
    print("Olá " + nome_da_pessoa1 + " e " + nome_da_pessoa2)
saudar2("Jeff", "Talitha")
```

```
Olá Jeff e Talitha
```

- O que acontece com a chamada à função `saudar2()` caso ela seja chamada com um número inteiro, por exemplo `saudar2(5)` ?

In [43]:
```
saudar2(5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-ab66ebb07187> in <module>
----> 1 saudar2(5)

<ipython-input-42-2da3e8fcf4ef> in saudar2(nome_da_pessoa)
      1 # Exemplo de função com um parâmetro, mas sem retorno explícito
      2 def saudar2(nome_da_pessoa):
----> 3     print("Olá " + nome_da_pessoa)

TypeError: can only concatenate str (not "int") to str
```

A função `main()`

In [1]:
```
# Listagem

from math import sqrt

def hipotenusa(x, y):
    return sqrt(x ** 2 + y ** 2)

def main():
    a = float( input("a: ") )
    b = float( input("b: ") )
```

```
    print("Hipotenusa: ", hipotenusa(a, b) )

main()
```

```
Hipotenusa:  7.0710678118654755
```

Variáveis locais

As variáveis que forem definidas dentro de uma função só existem enquanto ela estiver sendo executada.

In [60]:
```
def teste1(val1):
    val1 = 5
    return val1

val1 = 10
val1 = teste1(val1)
print(val1)
```

```
5
```

Conversões de Tipos

```
int(x)       # Converte x em um inteiro, truncando floats em 0
int(x, b)    # Converte x dado uma base b em inteiro
float(x)     # Converte x em um ponto flutuante
str(x)       # Converte x em uma string (texto)
```

Exercícios

[1] Responda às seguintes questões:

(a) Descreva em suas próprias palavras a subtarefa que a função `hipotenusa(x, y)` realiza.

(b) Qual(is) linha(s) de código define(m) o corpo da função `main()` ?

(c) Existe um retorno dentro da função `main()` e qual valor a função retorna?

(d) Descreva o resultado se a linha com `return sqrt(x ** 2 + y ** 2)` da função `hipotenusa` for substituída pela linha:

```
hyp = sqrt(x ** 2 + y ** 2)
```

Explique o comportamento que você observar.

---

[2] A função `main()` contém variáveis locais? Em caso positivo, identifique-as; caso contrário, explique porque não.

---

[3] Modifique a função `main()` para chamar `hipotenusa(a, b)` antes do comando `print()` armazenando o resultado em uma variável local, e então, usar a variável local no comando `print()` . Discuta os prós e contras dessa nova abordagem.

---

[4] Utilizando o código do cálculo da hipotenusa, crie um código que calcule a área de uma círculo a partir de um raio. Utilize uma função `área()` e uma função `main()` .

---

[5] Escreva um código que peça ao usuário dois números na função `main()` e utilize uma função separada para o cálculo da média desses números.

---

[6] Escreva um código que peça ao usuário por uma temperatura em Fahrenheit e então calcule e imprima a temperatura correspondente em Celsius. Use uma função para fazer a conversão.

---

[7] Modifique o exercício anterior para também computar e mostrar a temperatura equivalente em Kelvin. Use uma outra função de conversão (ou duas).

---

[8] Escreva um programa `heart.py` que peça ao usuário sua idade em anos (*y*) e então estime seu batimento cardíaco máximo em batidas por minuto, usando a fórmula $208 - 0.7y$. Use funções apropriadas.

---

[9] Escreva um programa que peça ao usuário pelos lados de um triângulo (a, b, c) e então compute a área de um triângulo utilizando a fórmula de Heron (Procure a fórmula se não se lembrar). Utilize funções apropriadas.

---

[10] Escreva um programa que calcule o novo saldo de uma conta poupança se os juros forem computados anualmente. Peça ao usuário pelo valor do principal (*p*), taxa de juros (*r*), e número de anos (*t*), e então mostre o novo saldo. Use a fórmula: $p(1 + r)^t$ e use funções apropriadas.

Entre com a taxa de juros como decimais: por exemplo, uma taxa de 4% deve ser colocada como 0.04. Use seu programa para determinar o número de anos que se leva para dobrar o valor do principal com uma taxa de juros de 1.5% ao ano.