INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

Desenvolvido por:
Rafael Puyau
Marcus Azevedo

Python - Aula 06

Objetivos da aula:

- Modulos

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Definição

- **Módulo** : é um arquivo python, ou seja, um arquivo com extensão .py
- **Pacote** : é um diretório com um ou mais arquivos python

Entendendo um pouco mais...

Ao saírmos e entrarmos no interpretador **Python**, as definições ali criadas como funções e variáveis são perdidas.

Portanto, se quisermos escrever um programa maior, será mais eficiente usar um editor de texto / código para preparar as entradas para o interpretador, e executá-lo usando o arquivo como entrada.

Isto é conhecido como criar um script. Se o nosso programa se tornar ainda maior, ***é uma boa prática dividí-lo*** em arquivos menores, para facilitar a manutenção.

Também é preferível usar um arquivo separado para uma função que estamos criando e pretendemos usá-la em vários outros programas. Desta forma, evitamos fazer uma cópia da função em cada um dos programas / scripts que irão utilizar a função.

Para um melhor organização, você deve criar uma estrutura de diretórios que facilite seu trabalho e consequentemente sua manutenção.

Reserve um diretório específico para armazenar seus módulos. Como vimos há pouco, este diretório será conhecido como pacote.

Python - Aula 06

Definição

> **DICA** : não esqueça de colocar sempre nomes descritivos, tanto para diretórios, arquivos, módulos e variáveis em **Python**

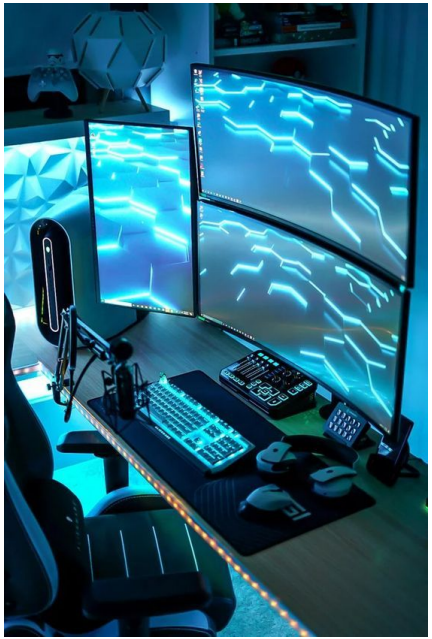
Dentro do pacote, você pode organizar seus arquivos da forma que achar melhor.

Exemplo 1:

```
meus_modulos
           |
           matematica.py
           caracteres.py
```

Exemplo 2:

```
meus_modulos
           |
           numeros
           |      |
           |       matematica.py
           |
           textos
                |
                caracteres.py
```



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

Como importar módulos no python

Há diversas formas de importarmos módulos built-in, bem como, nossos módulos. Nesta aula vamos criar 2 módulos - *matemática* e *caracteres*.

- **1ª forma**

  ```
  import caracteres
  ```

  ATENÇÃO : aqui vai uma advertência, pois quando importamos um módulo desta forma estamos trazendo TODAS as funções implementadas nele e, provavelmente, usaremos poucas funções.

  E o que acontece então? Você sobrecarrega a memória do seu computador desnecessariamente.

- **2ª forma**

  ```
  from caracteres import conta_vogais
  ```

  Para evitar carregar todo o módulo, podemos fazer a importação apenas da função que iremos utilizar.

Como importar os módulos no python

- **3ª forma**

```
from caracteres import (
        conta_vogais,
        conta_letras
        )
```

Quando queremos ou precisamos importar mais de uma função, devemos importar todas as funções dentro de uma tupla, seguindo o padrão de estilização de código do python - PEP 8.

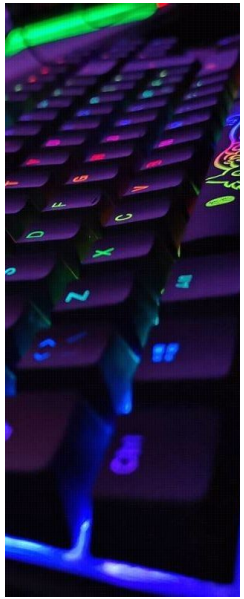
- **4ª forma**

```
from caracteres import *
```

Aqui, basicamente, importamos tudo que está no módulo *caracteres*. Praticamente como a primeira forma e já vimos que não faz muito sentido, além de carregar a memória do computador desnecessariamente.

Porém, vale ressaltar que desta forma basta chamarmos as funções diretamente pelo nome sem a necessidade de colocar o nome do módulo antes de suas chamadas:

```
from caracteres import *
conta_letras('Infinity School', espaco=True)
```



Python - Aula 06

Como importar os módulos no python

DICA : podemos "dar apelidos" para nossos módulos e/ou funções. Veja nestes exemplos:

- *Exemplo 1* `import caracteres as ch`

  Aqui demos o apelido de **ch** ao módulo *caracteres* e será com ele que chamaremos as funções: `ch.conta_vogais('Rafael')`

- *Exemplo 2* `from caracteres import conta_vogais as cv`

ATENÇÃO : para importar um ou mais módulos que não estejam no mesmo diretório que seu script, você deve especificar o caminho completo do pacote que contém os módulos que se deseja importar.

- *Exemplo*

```
meus_modulos
           |
            matematica.py
            caracteres.py
meu_programa.py
```

Considerando a estrutura de pastas acima, você deve importar os módulos desta forma:

- `from meus_modulos.matematica import nome_da_função_desejada`
- `from meus_modulos.caracteres import nome_da_função_desejada`

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

O so cláusula condicional nos módulos

Para evitarmos que nossos módulos sejam chamados diretamente pelo python e executados, devemos inserir ao final de cada arquivo uma cláusula condicional utilizando os *dunder objects* - `__name__` e `__main__` - que você acabou de aprender na seção anterior

Exemplo

```
if __name__ == '__main__':
    print(f'{__name__} é um módulo e não deve ser executado como um script')
```

Esta cláusula verifica se o nome do arquivo é igual ao objeto especial `__main__`.

Se for verdadeiro, emitimos uma mensagem na saída padrão informando que este arquivo é um módulo.

Arquivos de módulo não devem ser executados diretamente, e sim, devem ser chamados através de outros arquivos python - scripts.

Arquivos Python “compilados”

Para acelerar o carregamento de módulos, o Python guarda versões compiladas de cada módulo no diretório `__pycache__` com o nome do `modulo.versão.pyc`, onde a versão corresppponde ao formato do arquivo compilado; geralmente contêm o número da versão Python utilizada.

Esta convenção de nomes permite a coexistência de módulos compilados de diferentes releases e versões de Python.

O Python verifica a data de modificação do arquivo fonte mediante a versão compilada, para ver se está desatualizada e precisa ser recompilada.

É um processo completamente automático. Além disso, os módulos compilados são independentes de plataforma, portanto a mesma biblioteca pode ser compartilhada entre sistemas de arquiteturas diferentes.

Type Hint (Functions Annotations)

O *Python Moderno* introduziu as *typing hints* ou *annotations* que indica **APENAS** qual é o tipo de dados esperado ou retornado por variáveis ou função.

Como estamos falando de *Funções* nesta aula, vamos focar nas anotações para este bloco de código.

Vamos refatorar as duas versões da função `saudacao()`

Python - Aula 06

Sem parâmetro

```
[ ]  def saudacao() -> None:
       '''Imprime a mensagem de Boa tarde'''
       print('Boa tarde')
```

Repare que a função não retorna nada, logo especificamos que o retorno da mesma será do tipo `None`

Com parâmetro

```
[ ]  def saudacao(nome: str) -> str:
       '''Retorna uma string formatada e personalizada'''
       return f'Olá {nome}'
```

Agora, a função possui um parâmetro chamado `nome` que espera receber uma `string` e esta função retornará uma `string` formatada.

Python - Aula 06

Parâmetros

Posicionais

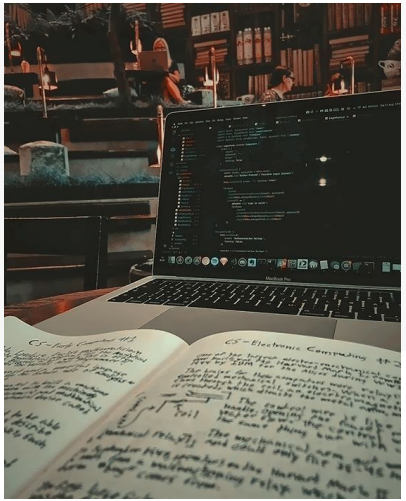
Simplesmente seguem a ordem em que são declarados, logo, na chamada da função os argumentos devem ser passados na mesma ordem em que os parâmetros aparecem.

OBS: Devemos fornecer / informar os argumentos na mesma quantidade de parâmetros que função possuir.

```
[ ]  def imprime_nome(nome: str, idade: int) -> str:
       '''Imprime o nome e a idade de uma pessoa

       params:
         nome - uma string literal
         idade - um inteiro positivo

       return:
         Uma string formatada com o nome e a idade da pessoa
       '''
       return f'{nome.title()} tem {idade} anos'
```



INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Parâmetros

Mais exemplos:

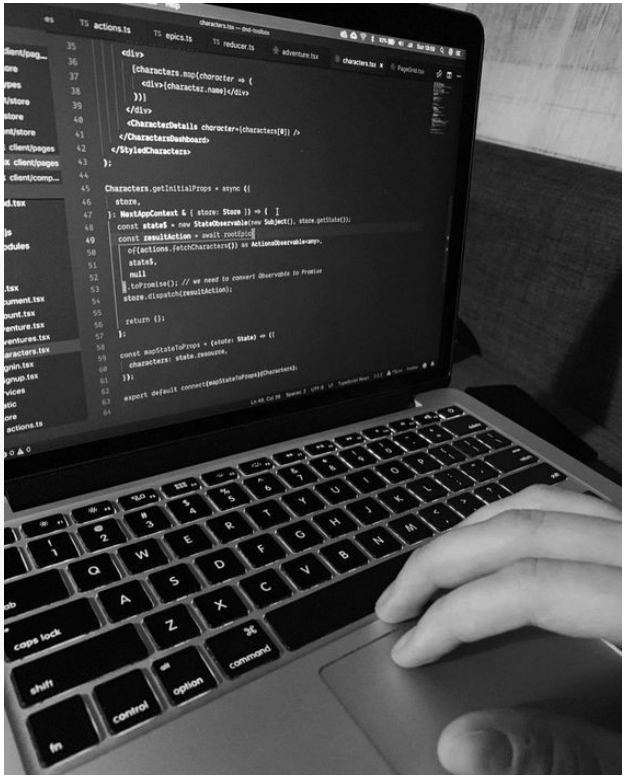
Descomente as linhas para poder testar

```
[ ]  # Veja o que é impresso
     # imprime_nome('rafael', 46)

     # Veja o que acontece
     # imprime_nome('rafael')

     # Veja o que acontece quando atribuímos o retorno
     # de uma função à uma variável
     # print(imprime_nome('Aluno', 18))
     # aluno = imprime_nome('Rute', 56)
     # print(type(aluno))
     # print(aluno)
```

```
<class 'str'>
Rute tem 56 anos
```



INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Parâmetros

Mais exemplos

Realizando 3 chamadas à função `imprime_nome`, mas passando argumentos diferentes

```
[ ] print(imprime_nome('Rafael', 46))
    print(imprime_nome('Allan', 31))
    print(imprime_nome('Beatriz', 11))
```

```
Rafael tem 46 anos
Allan tem 31 anos
Beatriz tem 11 anos
```

```
[ ] def soma(parcela1: int, parcela2: int) -> int:
      '''Retorna a soma de 2 números

      param:
        parcela1: número inteiro
        parcela2: número inteiro

      return:
        número inteiro'''
      return parcela1 + parcela2
```

```
[ ] print(soma.__doc__)
```

```
Retorna a soma de 2 números

      param:
        parcela1: número inteiro
        parcela2: número inteiro

      return:
        número inteiro
```

```
[ ] # resultado = soma(3, 4)
    print(f'O resultado da soma foi {soma(3, 4)}')
```

```
O resultado da soma foi 7
```

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Funções com retornos múltiplos

Uma função pode retornar mais de um valor que neste caso será uma **tupla**.

Já vimos em aula anterior como trabalhar com tuplas e aprendemos a desempacotar seus valores.

Veja o exemplo abaixo:

```
[ ]  def situacao_do_aluno(nota1: float, nota2: float) -> tuple:
       media = (nota1 + nota2) / 2
       situacao = 'Aprovado' if media > 6.99 else 'Reprovado'

       return media, situacao
```

Descomente as linhas para ver o detalhamento do que está acontecendo

```
[ ]  print(situacao_do_aluno(7, 8))
     print(type(situacao_do_aluno(7, 8)))
     media_aluno, situacao_academica = situacao_do_aluno(7, 8)
     print(f'O aluno A obteve média {media_aluno:.1f} e está {situacao_academica.lower()}')
```

```
(7.5, 'Aprovado')
<class 'tuple'>
O aluno A obteve média 7.5 e está aprovado
```

```
[ ]  for media, situacao in [situacao_do_aluno(7, 8)]:
       print(media, situacao, sep=' --- ')
```

```
7.5 --- Aprovado
```

```
[ ]  media_aluno, situacao_academica = situacao_do_aluno(5, 8)
     print(f'O aluno B obteve média {media_aluno:.1f} e está {situacao_academica.lower()}')
```

```
O aluno B obteve média 6.5 e está reprovado
```

IN
INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Palavras reservadas

def

É a palavra reservada da linguagem **Python** que define uma função

pass ou ...

Caso precise definir a assinatura de uma função, ou seja, seu cabeçalho, e definir o corpo da função mais tarde, você deverá usar a palavra reservada **pass** ou os **...**

```
[ ]  def soma(parcela1: int, parcela2: int) -> int:
       pass
```

```
[ ]  def calcula_imposto(valor: float, percentual: float) -> float:
       ...
```

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Palavras reservadas

return

Quando encontrada, encerra a função naquele momento retornando o valor ou valores para o mesmo ponto em que foi chamada.

ATENÇÃO : se houver código após o **return** este não será executado. Se estiver dentro de um bloco condicional, poderá fazer sentido, mas se não estiver você terá um erro - `Unreacheable code`

```
[ ]  def saudacao(nome: str) -> str:
       return f'Olá {nome}'
       print('Seja bem-vindo(a)') # unreacheable code
```

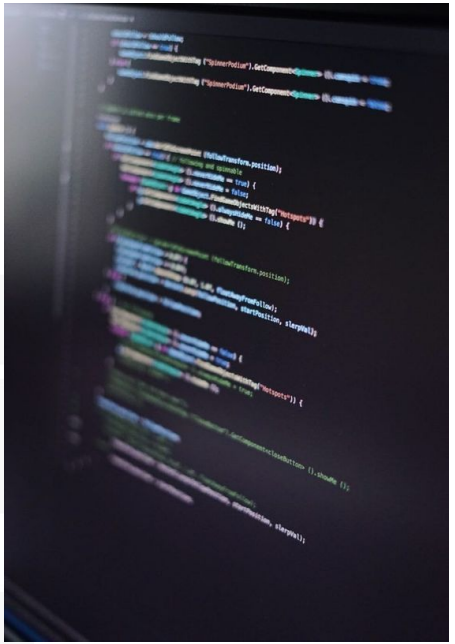
```
[ ]  def par_ou_impar(numero: int) -> int:
       if numero % 2 == 0:
         return 'PAR'
       else:
         return 'ÍMPAR'
```

Código acima refatorado para a **forma pythônica**

```
[ ]  def par_ou_impar(numero: int) -> int:
       if numero % 2 == 0:
         return 'PAR'
       return 'ÍMPAR'
```

```
[ ]  print(f'O número 8 é {par_ou_impar(8)}')
     print(f'O número 7 é {par_ou_impar(7)}')
```

```
O número 8 é PAR
O número 7 é ÍMPAR
```



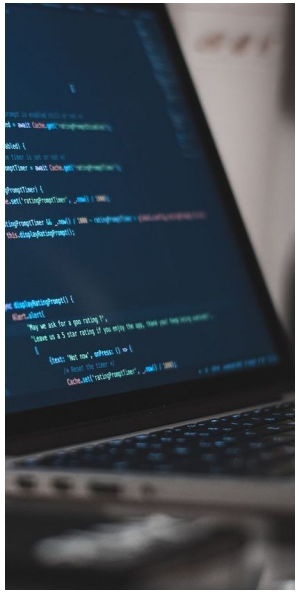
INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Hora de praticar

- IMC = peso / altura ** 2
  Crie uma função para calcular o IMC de 4 pessoas.
  Atenção: Use as seguintes estruturas:
  - laço de repetição.
  -listas
  -zip
- Faça uma função para calcular o valor/hora de um funcionário.
- Faça uma função que retorne quantas letras possui uma palavra.

  Se for passado uma frase, a função deverá retornar o número de letras, espaços vazios e quantos sinais de pontuação.



Obs: Você pode fazer o download do notebook desta aula que encontra-se na sessão de downloads do portal.
Neste arquivo você encontrará o gabarito destas questões.

IN
INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Python - Aula 06

Hora de praticar

Faça um programa que onde o usuário deverá informar qual operação ele deseja realizar através dos sinais dessas operações, ou seja:

- `+` para soma
- `-` para subtração
- `*` para multiplicação
- `/` para divisão

E depois informe 2 números inteiros.

Atenção:

- use bloco condicional para chamar a função apropriada
- crie 4 funções das operações matemáticas básicas que retornem seus resultados
- crie docstring para cada função
- utilize as *annotations* também
- a função de divisão deverá informar ao usuário uma mensagem de erro se o *divisor* for igual a zero

Obs: Você pode fazer o download do notebook desta aula que encontra-se na sessão de downloads do portal.
Neste arquivo você encontrará o gabarito destas questões.

Você concluiu a aula 06 do seu módulo de Python.
Continue praticando e até a próxima aula!

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER