07 {..

Modularização e Pacotes

..}

Formador: Ricardo Mourão

Modularização e Pacotes

```
def analisa_triangulo(r1, r2, r3):
    if r1 < r2 + r3 and r2 < r1 + r3 and r3 < r1 + r2:
        print(“Os segmentos acima podem formar um triangulo”, end=“ “)
        if r1 == r2 == r3:
            print(“EQUILÁTERO.”)
        elif r1 != r2 != r3 != r1:
             print(“Escaleno.”)
        else:
             print(“Isósceles.!”)
    else:
        print(“Os segmentos acima não podem formar um triângulo.”)
```

Formador: Ricardo Mourão

Modularização e Pacotes

```
def analisa_triangulo(r1, r2, r3):
    if r1 < r2 + r3 and r2 < r1 + r3 and r3 < r1 + r2:
        print(“Os segmentos acima podem formar um triangulo”, end=“ “)
        if r1 == r2 == r3:
            print(“EQUILÁTERO.”)
        elif r1 != r2 != r3 != r1:
             print(“Escaleno.”)
        else:
             print(“Isósceles.!”)
    else:
        print(“Os segmentos acima não podem formar um triângulo.”)

seg1 = float(input(“primeiro segmento: “))
seg2 = float(input(“segundo segmento: “))
seg3 = float(input(“terceiro segmento: “))
analisa_triangulo(seg1, seg2, seg3)
```

Formador: Ricardo Mourão

Modularização e Pacotes

```
seg1 = float(input(“primeiro segmento: “))
seg2 = float(input(“segundo segmento: “))
seg3 = float(input(“terceiro segmento: “))
analisa_triangulo(seg1, seg2, seg3)
```

functions.py

```
def analisa_triangulo(r1, r2, r3):
    if r1 < r2 + r3 and r2 < r1 + r3 and r3 < r1 + r2:
        print(“Os segmentos acima podem formar um triangulo”, end=“ “)
        if r1 == r2 == r3:
            print(“EQUILÁTERO.”)
        elif r1 != r2 != r3 != r1:
            print(“Escaleno.”)
        else:
            print(“Isósceles.!”)
    else:
        print(“Os segmentos acima não podem formar um triângulo.”)
```

Formador: Ricardo Mourão

Modularização e Pacotes

```
import functions
seg1 = float(input(“primeiro segmento: “))
seg2 = float(input(“segundo segmento: “))
seg3 = float(input(“terceiro segmento: “))
functions.analisa_triangulo(seg1, seg2, seg3)
```

functions.py

```
def analisa_triangulo(r1, r2, r3):
    if r1 < r2 + r3 and r2 < r1 + r3 and r3 < r1 + r2:
        print(“Os segmentos acima podem formar um triangulo”, end=“ “)
        if r1 == r2 == r3:
            print(“EQUILÁTERO.”)
        elif r1 != r2 != r3 != r1:
            print(“Escaleno.”)
        else:
            print(“Isósceles.!”)
    else:
        print(“Os segmentos acima não podem formar um triângulo.”)
```

Formador: Ricardo Mourão

Modularização e Pacotes

```python
from functions import analisa_triangulo
seg1 = float(input(“primeiro segmento: “))
seg2 = float(input(“segundo segmento: “))
seg3 = float(input(“terceiro segmento: “))
analisa_triangulo(seg1, seg2, seg3)
```

functions.py

```python
def analisa_triangulo(r1, r2, r3):
    if r1 < r2 + r3 and r2 < r1 + r3 and r3 < r1 + r2:
        print(“Os segmentos acima podem formar um triangulo”, end=“ “)
        if r1 == r2 == r3:
            print(“EQUILÁTERO.”)
        elif r1 != r2 != r3 != r1:
            print(“Escaleno.”)
        else:
            print(“Isósceles.!”)
    else:
        print(“Os segmentos acima não podem formar um triângulo.”)
```

Formador: Ricardo Mourão

Modularização e Pacotes

```
from functions import analisa_triangulo

from random import randint

from datetime import datetime

from time import sleep
```

Formador: Ricardo Mourão

Modularização e Pacotes

{ Isto é o conceito de modularização

Modularização é a prática de dividir um programa em módulos separados, cada um com funcionalidades específicas.

Pode pensar num módulo como um ficheiro Python (.py) que contém definições de funções, classes e variáveis, assim como código executável.

Formador: Ricardo Mourão

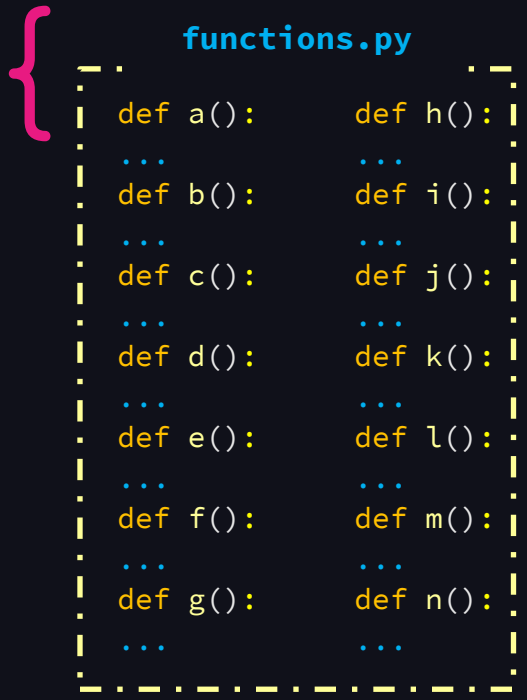
Modularização e Pacotes

{ Isto é o conceito de modularização

- **Organização**: Ajuda a organizar melhor o código. Em vez de ter todo o código num único ficheiro, ele é dividido em diferentes partes, em módulos separados.
- **Reutilização**: Permite a reutilização de código. Pode usar funções e classes definidas num módulo em vários projetos sem ter que copiar e colar o código.
- **Manutenção**: Facilita a manutenção. Ao modificar um módulo, não é necessário alterar o programa inteiro, apenas a parte específica.
- **Colaboração**: Torna mais fácil para várias pessoas trabalharem no mesmo projeto. Cada um pode focar-se num módulo específico.

Formador: Ricardo Mourão

Modularização e Pacotes

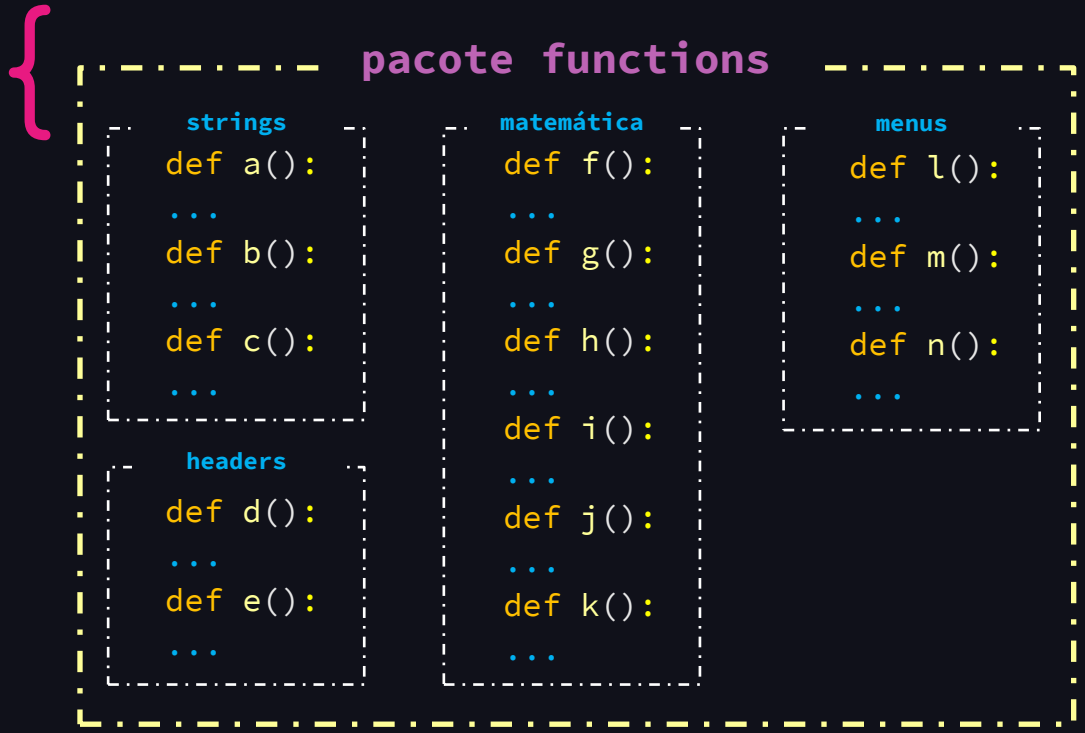


Pacote

```
import functions
```

Formador: Ricardo Mourão

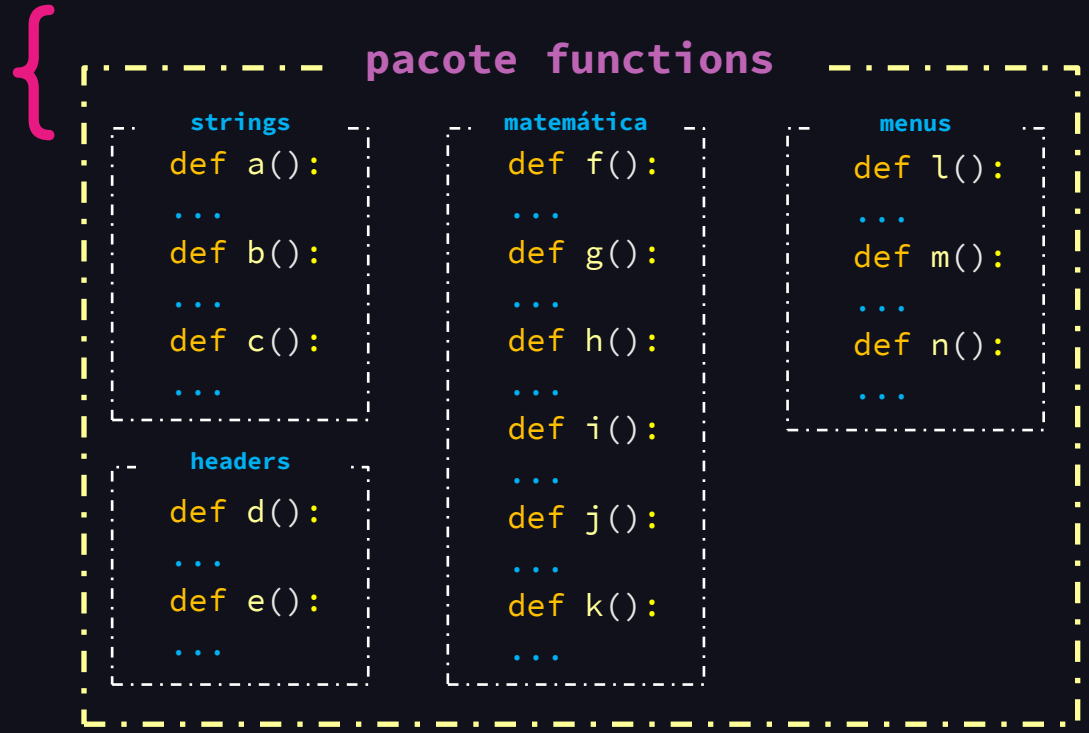
Modularização e Pacotes

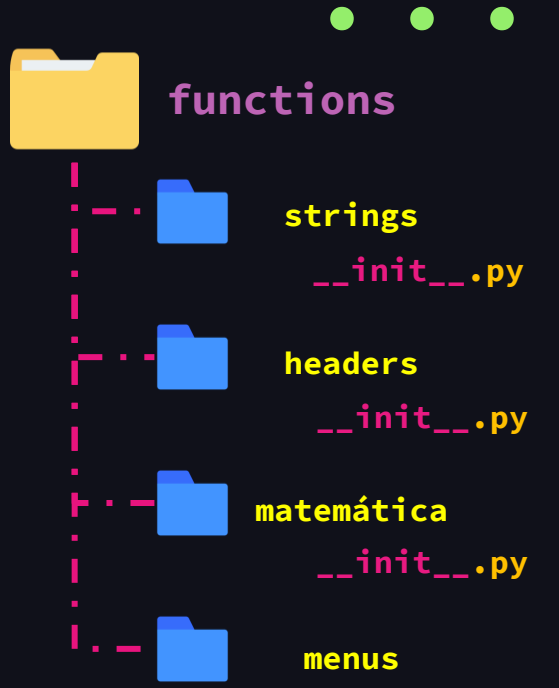



```
import functions

from functions import menus

from functions import menus, strings
```

Formador: Ricardo Mourão

Modularização e Pacotes





Formador: Ricardo Mourão

08 {..

Erros e exceções

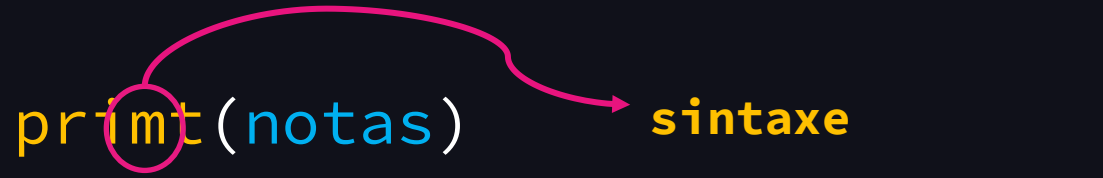
..}

Formador: Ricardo Mourão

Erros e exceções

notas.py

```
primt(notas)
```



notas.py

```
print(notas)
```

Formador: Ricardo Mourão

Erros e exceções

notas.py

```python
notas = list()
while True:
    nota = int(input("Digite uma nota: "))
    notas.append(nota)
    opcao = input("Quer continuar? [S/N]").strip().lower()
    if opcao == "n":
        break
print(notas)
```

E se o utilizador digitar 9.5?

Formador: Ricardo Mourão

Erros e exceções

```
Digite uma nota: 9.5
Traceback (most recent call last):
  File "C:\Users\ricar\Documents\Aulas Python IEFP\Aula\teste.py", line 3, in <module>
    nota = int(input("Digite uma nota: "))
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: '9.5'

Process finished with exit code 1
```

Formador: Ricardo Mourão

Erros e exceções

divisao.py

```
def divisao(a, b):
    return a/b



num1 = int(input("Digite um número: "))
num2 = int(input("Digite outro número: "))
print(divisao(num1, num2))
```

num1 = 10
num2 = 0

Formador: Ricardo Mourão

Erros e exceções

```
Digite um número: 10
Digite outro número: 0
Traceback (most recent call last):
  File "C:\Users\ricar\Documents\Aulas Python IEFP\Aula\teste.py", line 7, in <module>
    print(divisao(num1, num2))
          ^^^^^^^^^^^^^^^^^^^
  File "C:\Users\ricar\Documents\Aulas Python IEFP\Aula\teste.py", line 2, in divisao
    return a/b
           ~^~
ZeroDivisionError: division by zero

Process finished with exit code 1
```

Formador: Ricardo Mourão

Erros e exceções

Exceção	Causa
AssertionError	Raised when an assert statement fails.
AttributeError	Raised when attribute assignment or reference fails.
EOFError	Raised when the input() function hits end-of-file condition.
FloatingPointError	Raised when a floating point operation fails.
GeneratorExit	Raise when a generator's close() method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when the index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.

Formador: Ricardo Mourão

Erros e exceções

{

Exceção	Causa
KeyboardInterrupt	Raised when the user hits the interrupt key (Ctrl+C or Delete).
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in local or global scope.
NotImplementedError	Raised by abstract methods.
OSError	Raised when system operation causes system related error.
OverflowError	Raised when the result of an arithmetic operation is too large to be represented.
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.
RuntimeError	Raised when an error does not fall under any other category.

Formador: Ricardo Mourão

Erros e exceções

{

Exceção	Causa
StopIteration	Raised by next() function to indicate that there is no further item to be returned by iterator.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
TabError	Raised when indentation consists of inconsistent tabs and spaces.
SystemError	Raised when interpreter detects internal error.
SystemExit	Raised by sys.exit() function.
TypeError	Raised when a function or operation is applied to an object of incorrect type.
UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
UnicodeEncodeError	Raised when a Unicode-related error occurs during encoding.
UnicodeDecodeError	Raised when a Unicode-related error occurs during decoding.
UnicodeTranslateError	Raised when a Unicode-related error occurs during translating.
ValueError	Raised when a function gets an argument of correct type but improper value.
ZeroDivisionError	Raised when the second operand of division or modulo operation is zero.

Formador: Ricardo Mourão

Erros e exceções

{

EXCEPTION

Formador: Ricardo Mourão

Erros e exceções

```
try:    #é usado para acautelar um código que pode causar uma exceção ou erro.
    bloco

except:  #captura a exceção que foi levantada no bloco try
    se falhou

else:     #é executado apenas se nenhuma exceção for levantada no bloco try.
    se funcionou

finally:    #é executado independentemente de uma exceção ser levantada ou não
    executa sempre
```

opcionais

Formador: Ricardo Mourão

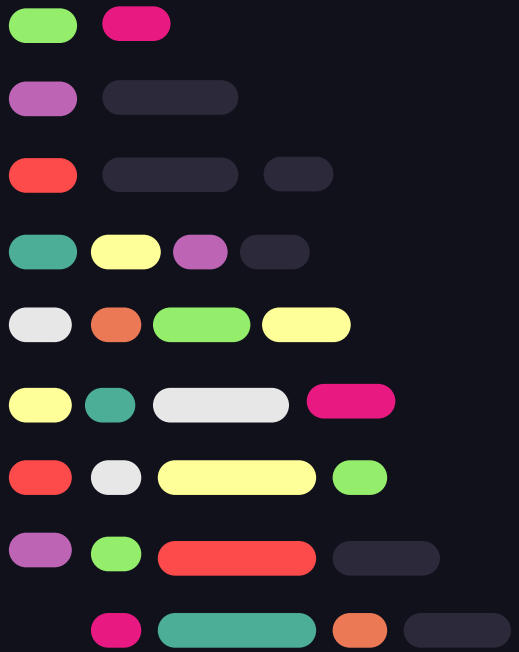
Erros e exceções

escreve.py

```
file = None
try:
    file = open('exemplo.txt', 'w')
    file.write(“Olá, estou num ficheiro TXT\n")
    file.write("Este é um exemplo de manipulação de arquivos em Python.\n")
except:
    print("Erro ao abrir ou escrever no arquivo.")
else:
    print("Arquivo escrito com sucesso.")
finally:
    if file:
        file.close()
        print("Arquivo fechado.")
```

Formador: Ricardo Mourão

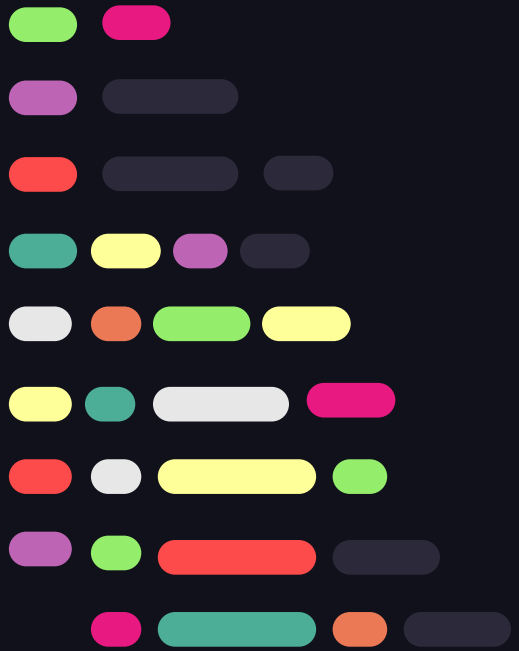
PRÁTICA! Exercicio 37



Escreva um programa que peça ao utilizador para inserir dois números e divida o primeiro pelo segundo. Utilize o tratamento de exceções para lidar com casos em que o segundo número é zero e quando a entrada não é um número válido.

Formador: Ricardo Mourão

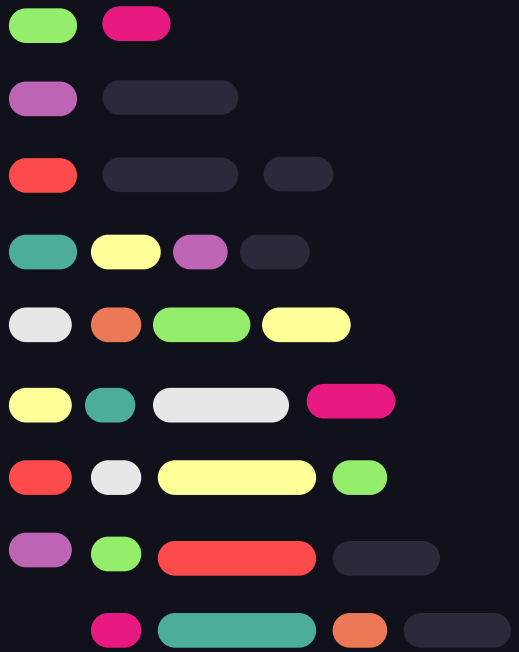
PRÁTICA! Exercicio 38



Crie um código em Python que teste se o site da Castleform está acessível a partir do seu computador.

Formador: Ricardo Mourão

PRÁTICA! Exercicio 39



Com recurso às novas técnicas de desenvolvimento aprendidas, crie:

```
--- Calculadora ---
[ 1 ] - Tabuada
[ 2 ] - Calculadora
[ 3 ] - Números Pares
[ 4 ] - Sair
```

Mediante a opção solicitada o sistema deve executar a ação do menu.

Formador: Ricardo Mourão