SENAC HUB ACADEMY
CENTRO DE INOVAÇÃO EM EDUCAÇÃO PROFISSIONAL

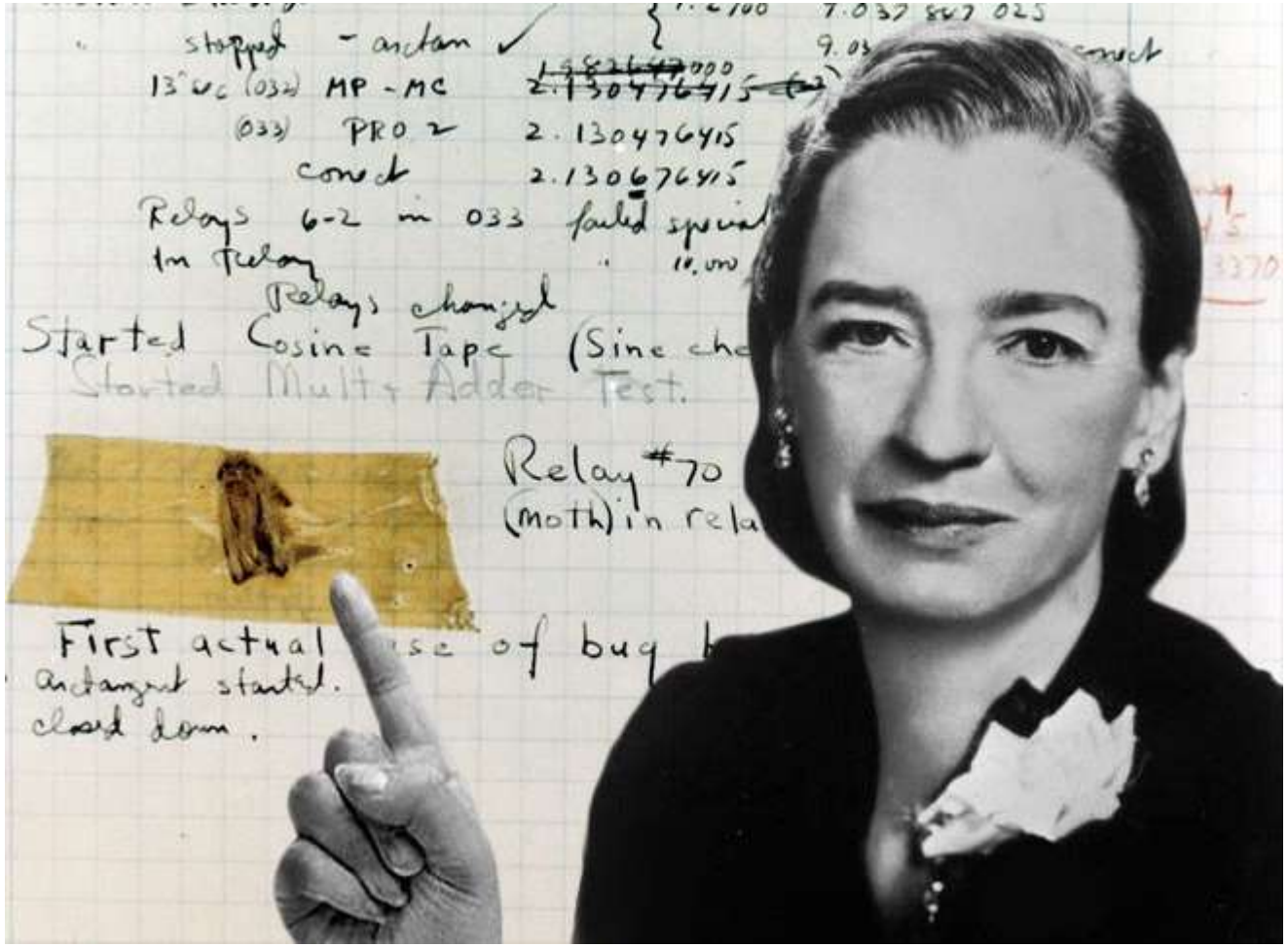
UC 3: DESENVOLVER ALGORITMOS

PROFESSOR: THIAGO ALMEIDA

BUGS

- Um bug é um erro no código que faz com que um programa se comporte de maneira inesperada ou falhe ao executar uma função.

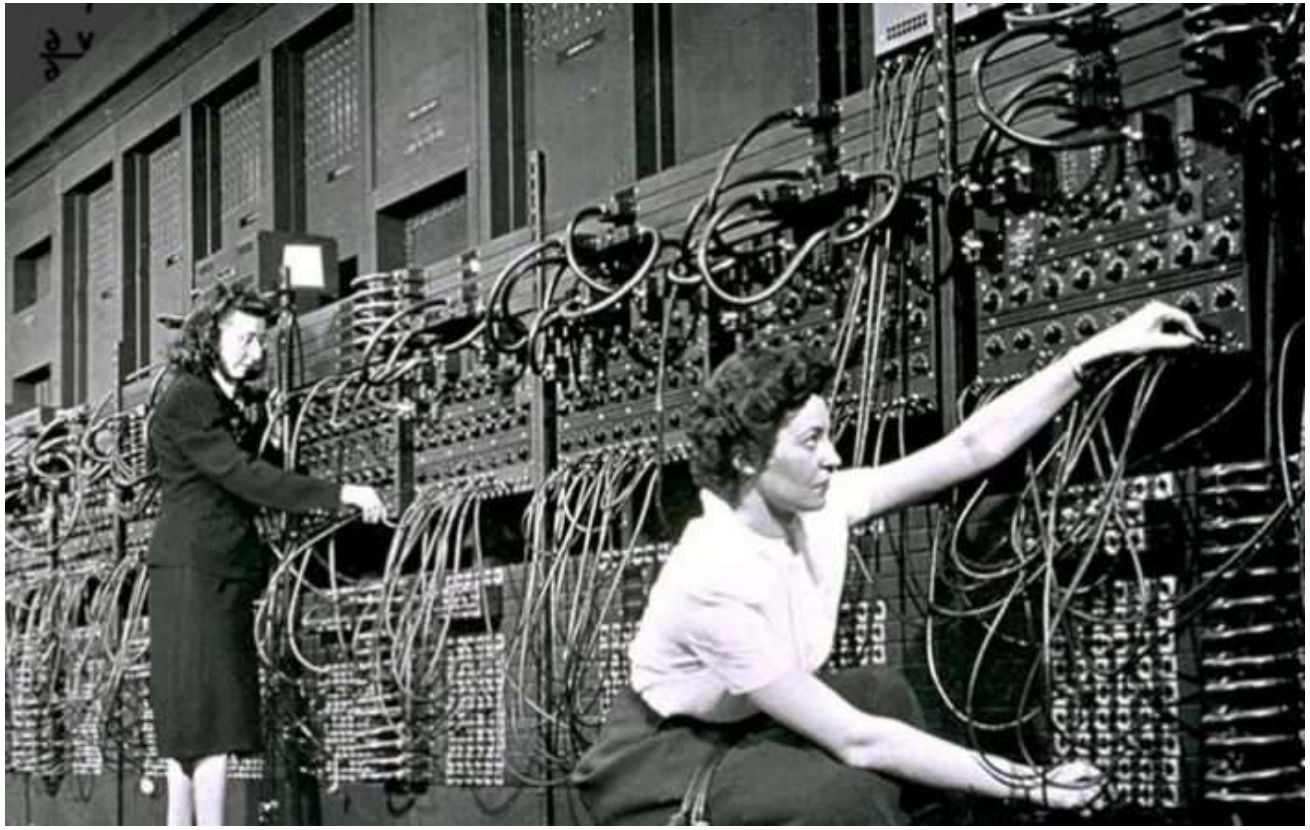
- A origem do termo é frequentemente atribuída a Grace Hopper, pioneira da computação, que encontrou uma mariposa causando falhas em seu computador.



The Origin of the Term 'Computer Bug'

ENIAC (1946 – 1955)

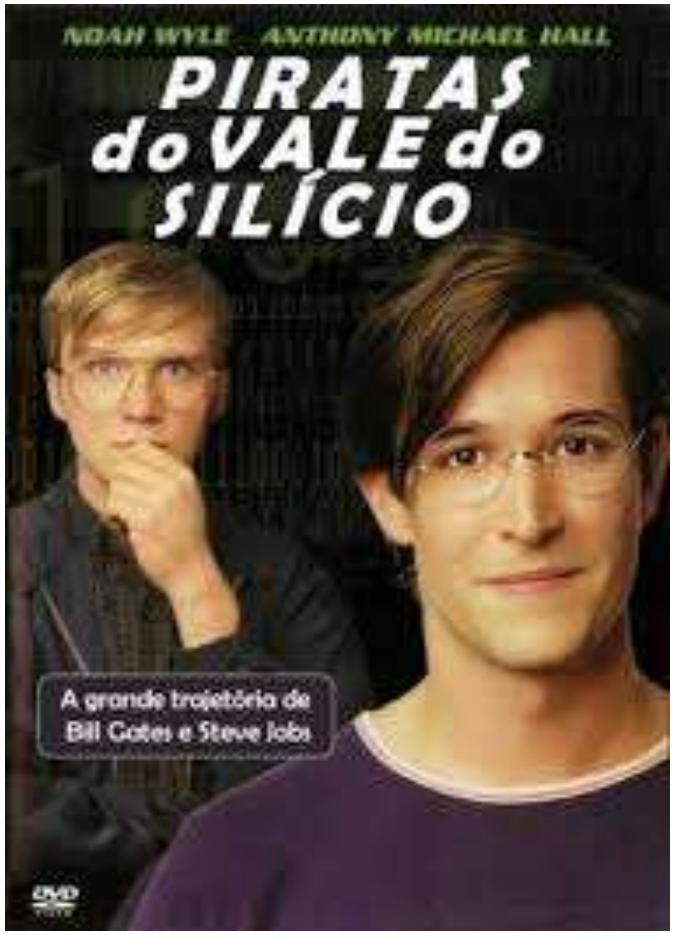
O primeiro computador do mundo, funcionava por meio de circuitos e válvulas eletrônicas. Desenvolvido a pedido do exército dos Estados Unidos, o ENIAC era um monstrengo de 30 toneladas de peso que ocupava uma área de 180 m² de área construída. A máquina contava com um hardware equipado com 70 mil resistores e 18 mil válvulas de vácuo que em funcionamento consumiam cerca de 200 mil watts de energia.



Eletronic Numerical Integrator and Computer

DICAS SOBRE A HISTÓRIA DA COMPUTAÇÃO

- A Rede Social - (Facebook)
- Jogo da Imitação
- Jobs (Netflix)
- O Código Bill Gates (Netflix)
- O Quinto Poder
- Matrix
- Jogos de Guerra
- Invasores
- Snowden
- Revolution OS (Linux - Youtube)
- Som na Faixa (Série sobre o Spotify Netflix)
- Piratas do Vale do Silício (Apple vs Microsoft)

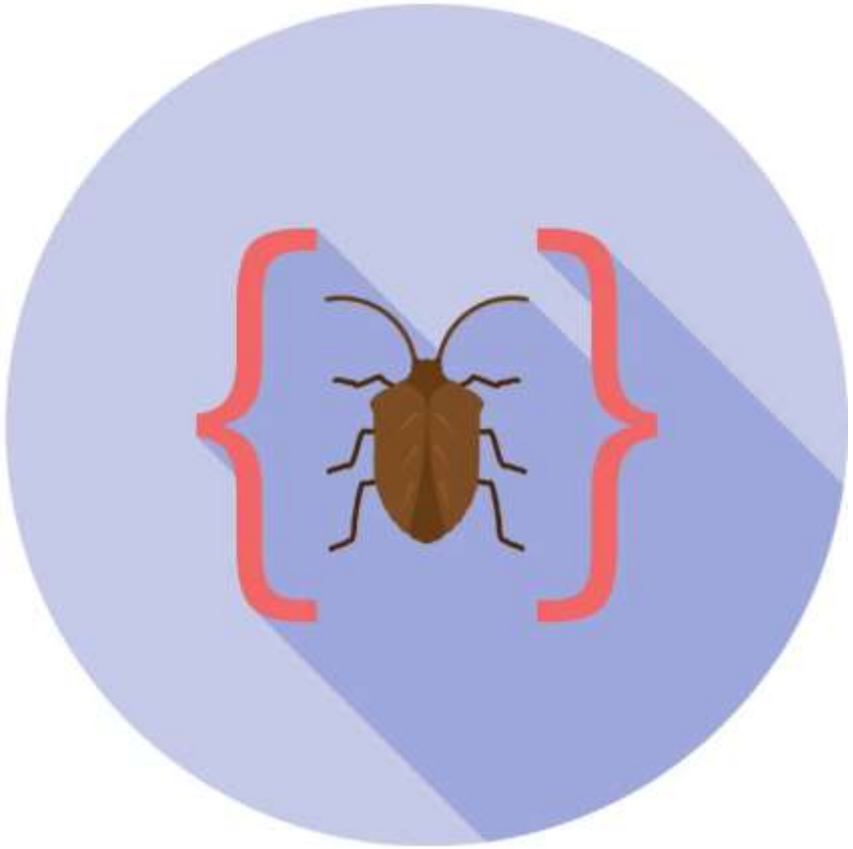


PRINCIPAIS ERROS EM CÓDIGOS

- Um evento que interrompe o fluxo normal de processamento de uma tarefa.

Causa o término inesperado de um programa:

- Problemas no hardware, arrays fora de índice;
- Valores de variáveis;
- Divisão por zero;
- Parâmetros de métodos, falha de Memória;
- Erro de entrada e saída (I/O);
- Erros da aplicação;
- Saldo insuficiente;
- Usuário não existe;
- Nota inválida;



ERROS DE SINTAXE PYTHON

Erros de sintaxe são comuns em qualquer linguagem de programação, e Python não é exceção. Quando escrevemos código em Python, é fundamental ter atenção à sintaxe correta, pois qualquer erro pode levar a falhas na execução do programa.

Por exemplo, não escrever : no final da linha linha com um def produz uma mensagem de certa forma redundante SyntaxError: invalid syntax.

```
print("Hello, world!"
```

```
  Input In [1]
    print("Hello, world!"
          ^
SyntaxError: '(' was never closed
```

```
>>> while True print('Hello world')
  File "<stdin>", line 1
    while True print('Hello world')
               ^^^^^
SyntaxError: invalid syntax
```

ERROS DE EXECUÇÃO PYTHON

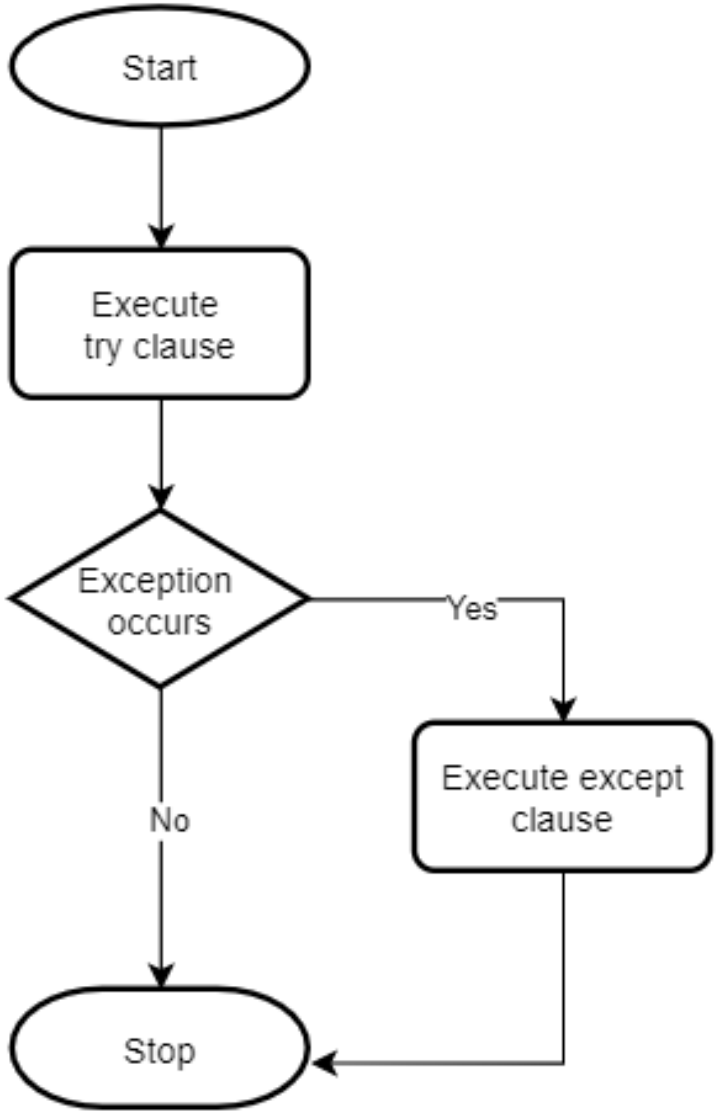
Mesmo que uma instrução ou expressão seja sintaticamente correta, ela poderá causar um erro quando for feita uma tentativa de executá-la. Erros detectados durante a execução são chamados de exceções e não são totalmente fatais.

A maioria das exceções não é tratada pelos programas e resulta em mensagens de erro, conforme mostrado a imagem mostrada ao lado:

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

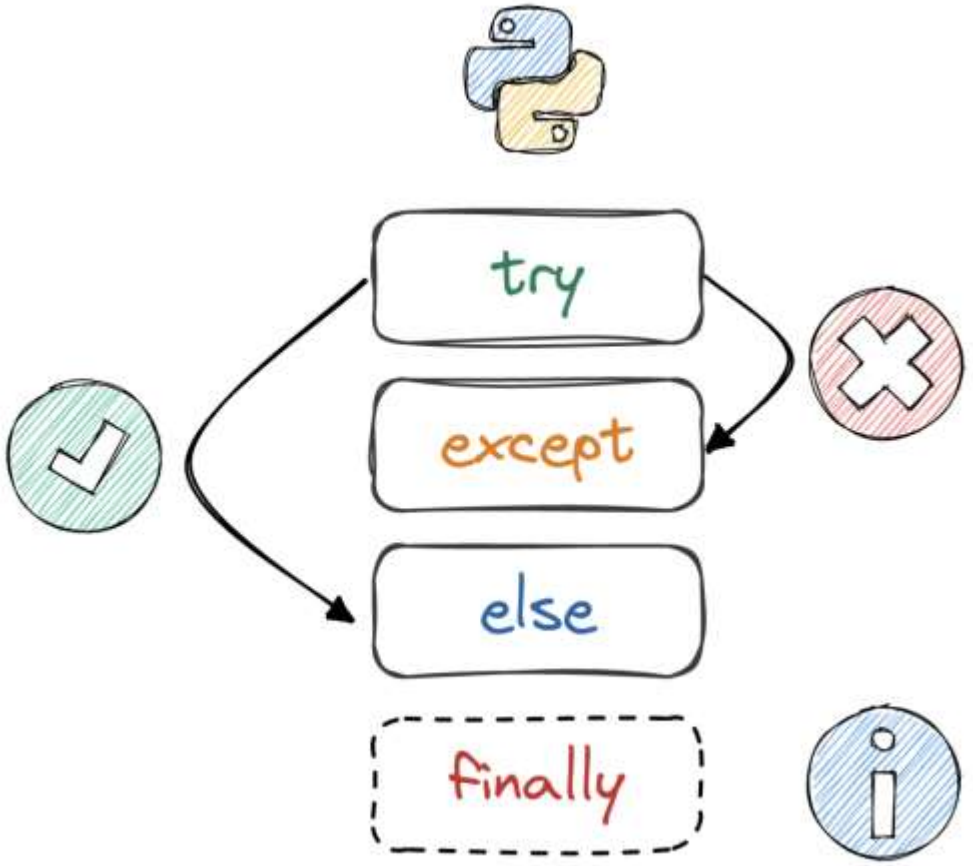
BLOCO TRY

- O bloco try é usado para envolver o código suscetível a erros. O Python tenta executar o código dentro deste bloco, e se um erro ocorre, a execução é interrompida e o controle é transferido para o bloco catch mais próximo.



BLOCO EXCEPT

- Após a captura de um erro pelo bloco try, o bloco except é executado, permitindo que o erro seja tratado de forma adequada. O bloco catch recebe um argumento, geralmente denominado error, que contém informações sobre o erro ocorrido.



BLOCO FINALLY

- Opcionalmente, o bloco finally pode ser utilizado após os blocos try e except. Este bloco será executado independentemente do resultado anterior, sendo ideal para realizar limpezas ou finalizar processos que devem ser concluídos após a tentativa de execução do código, seja ela bem-sucedida ou não.

```
try:
    f = open("demofile.txt")
    try:
        f.write("Lorum Ipsum")
    except:
        print("Something went wrong when writing to the file")
    finally:
        f.close()
except:
    print("Something went wrong when opening the file")
```

TRY / EXCEPT EXAMPLE

- Usado para o tratamento de exceção da função divide, previne apresentar erro para o usuário, se o segundo parâmetro for Zero, apresenta uma mensagem personalizada.

```
13  # Python code to illustrate
14  # working of try()
15  def divide(x, y):
16      try:
17          # Floor Division : Gives only Fractional Part as Answer
18          result = x // y
19          print("Yeah ! Your answer is :", result)
20      except ZeroDivisionError:
21          print("Sorry ! You are dividing by zero ")
22
23  # Look at parameters and note the working of Program
24  divide(3, 0)
```

TRY / EXCEPT EXAMPLE

```python
def fahrenheit_to_celsius(f):
    return (f - 32) * 5/9

while True:
    temp = input("Enter a temperature in Fahrenheit: ")

    try:
        fahrenheit_temp = float(temp)
        celsius_temp = fahrenheit_to_celsius(fahrenheit_temp)
        print(f"{fahrenheit_temp}°F is equal to {celsius_temp:.2f}°C.")
        break
    except ValueError:
        print("That's not a valid temperature. Please enter a number.")
```

EXERCÍCIOS

1. Escreva um programa Python que solicite ao usuário que insira um inteiro e gere uma exceção ValueError se a entrada não for um inteiro válido.

2. Escreva um programa Python que solicite ao usuário que insira dois números e gere uma exceção TypeError se as entradas não forem numéricas.

3. Crie uma função para realizar o saque de conta corrente. Uma exceção é lançada sempre que o saldo da conta for inferior ao valor sacado.

4. Faça um Programa que verifique se uma letra digitada é "F", "M" ou "L" conforme a letra escrever: F - Feminino, M - Masculino, L – LGBT. Caso o usuário não selecione a opção correta uma exceção apresenta a mensagem: “Opção Inválida”.

5. Crie uma função que receba e retorne a raiz quadrada de um número recebido por parâmetro, caso ocorra uma exceção de ValueError, apresentar a seguinte mensagem: "Valor impróprio para raiz quadrada“

RUNTIME ERROR

```python
try:
    x = int(input("Digite um número: "))
    y = int(input("Digite outro número: "))
    if y == 0:
        raise RuntimeError("Divisão por zero não é permitida.")
    resultado = x / y
    print(f"O resultado é: {resultado}")
except RuntimeError as e:
    print(f"Erro de execução: {e}")
except ValueError:
    print("Erro: Entrada inválida! Por favor, digite um número.")
```