Fatec
Rio Claro

Linguagem de Programação I

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler

"Qualquer tolo pode escrever código que um computador entenda. Bons programadores escrevem código que humanos conseguem entender."

Martin Fowler

Próximo Texto | Índice

CIÊNCIA

Sonda perde contato ao se aproximar de Marte

Nasa/Divulgação

A sonda Mars Climate Orbiter (MCO), em 98, que pode ter se perdido no momento em que entrava na órbita do planeta Marte

das agências internacionais

Depois de viajar por quase um ano para chegar a Marte, a sonda norte-americana Mars Climate Orbiter (MCO), de US$ 125 milhões, pode ter se perdido para sempre no momento em que entrava na órbita do planeta.

Um erro de navegação pode ter feito com que a sonda chegasse a apenas 60 km de Marte -100 km mais perto do que o planejado e 25 km abaixo do nível de segurança do projeto.

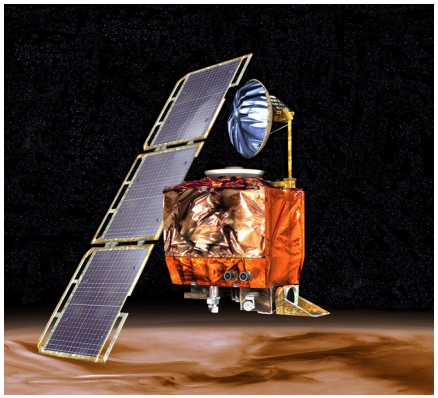
O erro de rota pode ter danificado o sistema de comunicação da MCO, ou ela pode ter se chocado contra o planeta.

"Não acreditamos que a nave tenha sobrevivido", disse John McNamee, gerente do projeto, à rede de notícias CNN.

"Parece que havia algo de errado com seu sistema de navegação."

Mars Climate Orbiter

Fatec
Rio Claro



Lançamento: 11 de dezembro de 1998

Falha: 23 de setembro de 1999

Motivo do fracasso: Erro de conversão de unidades

Causa específica: Um dos times da equipe (da Lockheed Martin, nos EUA) utilizou unidades imperiais (libras-força), enquanto o sistema da NASA e o padrão científico internacional utilizavam unidades métricas (newtons).

Consequência: A sonda entrou na atmosfera de Marte com uma altitude muito menor do que o planejado (aproximadamente 57 km ao invés dos 140-150 km), e se desintegrou ou colidiu com a superfície do planeta.

Funções

Funções

Fatec
Rio Claro

Funções são especialmente interessantes para isolar uma tarefa específica em um trecho do programa. Isso pemite que a solução de um problema seja reutilizada em outras partes do programa, sem precisar repetir as mesmas linhas.

args

Fatec
Rio Claro

Em Python, *args é uma convenção usada para permitir que uma função receba um número variável de argumentos posicionais.

```
def soma(n1, n2):

def soma(n1, n2, n3):

def soma(n1, n2, n3, n4):
```

args

Em Python, *args é uma convenção usada para permitir que uma função receba um número variável de argumentos posicionais.

```
def soma(n1, n2):
def soma(n1, n2, n3):
def soma(n1, n2, n3, n4):
```

```
def soma(*args):
    total = 0
    for numero in args:
        total += numero
    return total
```

args

Fatec
Rio Claro

O asterisco * antes do nome *args* indica que a função pode receber vários argumentos posicionais como uma tupla.

args é apenas um nome comum, você pode usar outro nome, mas *args* é o mais usado por convenção.

Parâmetros nomeados

Fatec Rio Claro

Quando especificamos o nome dos parâmetros, podemos passá-lo em qualquer ordem:

```
def retangulo(largula, altura, caracter='*')
    linha = caracter * largura
    for i in range(altura):
        print(linha)
```

kwargs

Fatec
Rio Claro

Em Python, **kwargs permite que uma função aceite um número variável de argumentos nomeados (ou seja, passados como chave=valor), e os armazena em um dicionário.

```
def exibir_info(**kwargs):
    for chave, valor in kwargs.items():
        print(f"{chave}: {valor}")
exibir_info(nome="Orlando", cidade="RC", emprego=”Fatec”)
```

kwargs

Fatec
Rio Claro

O duplo asterisco ** indica que a função irá receber vários argumentos com nomes (palavras-chave).

kwargs é uma abreviação de "keyword arguments".

O resultado será um dicionário com as chaves e valores dos argumentos nomeados.

Funções anônimas

Funções anônimas

Fatec
Rio Claro

É uma função sem nome, ou seja, você não precisa usar def.

Em Python, usamos a palavra-chave lambda para criar esse tipo de função.

São geralmente usadas para tarefas rápidas e simples, principalmente quando você precisa passar uma função como argumento.

Funções anônimas

Fatec
Rio Claro

A função lambda só pode ter uma única expressão, sem comandos múltiplos (como if, for, etc. em blocos separados).

lambda argumentos: expressão

Funções anônimas

Fatec
Rio Claro

```
# Função tradicional
def dobro(x):
    return x * 2


# Função lambda equivalente
dobro_lambda = lambda x: x * 2


print(dobro(5))        # 10
print(dobro_lambda(5)) # 10
```

Funções anônimas

Fatec
Rio Claro

```
alunos = [("Ana", 8), ("Carlos", 5), ("Bruna", 10)]

# Usando lambda para dizer: "ordene pelo segundo item"
ordenado = sorted(alunos, key=lambda aluno: aluno[1])

print(ordenado)
# Resultado: [('Carlos', 5), ('Ana', 8), ('Bruna', 10)]
```

Funções anônimas
Quando usar ?

Fatec
Rio Claro

- Precisa de uma função rápida e simples.

- Vai passar a função como argumento (ex: map, filter, sorted).

- Não precisa reutilizar a função depois (caso contrário, prefira def).

Funções anônimas
Quando não usar ?

Fatec
Rio Claro

- Quando a lógica é complexa ou envolve várias linhas.

- Quando a função precisa ser reutilizada com frequência.

- Quando você quer que o código seja muito legível para iniciantes.

Testes Unitários

Teste unitário

Fatec
Rio Claro

Um teste de unidade verifica uma unidade individual de código (como uma função) para garantir que ela se comporte como esperado em diferentes situações.

O objetivo é encontrar erros o mais cedo possível no desenvolvimento.

unittest

Fatec
Rio Claro

unittest é o módulo padrão de testes automatizados do Python.

Ele permite que você escreva testes de unidade para verificar se partes do seu código (geralmente funções ou métodos) estão funcionando corretamente.

https://docs.python.org/3/library/unittest.html

Vantagens do unittest

Fatec
Rio Claro

Automatiza os testes do seu código.

Facilita a manutenção e evolução de projetos.

Garante que alterações não quebrem funcionalidades existentes.

É integrado ao Python (não precisa instalar nada).

Métodos unittest

Fatec
Rio Claro

Método	Descrição
`assertEqual(a, b)`	Verifica se `a == b`
`assertNotEqual(a, b)`	Verifica se `a != b`
`assertTrue(x)`	Verifica se `x` é `True`
`assertFalse(x)`	Verifica se `x` é `False`
`assertIsNone(x)`	Verifica se `x is None`
`assertRaises(Exception)`	Verifica se uma exceção é levantada

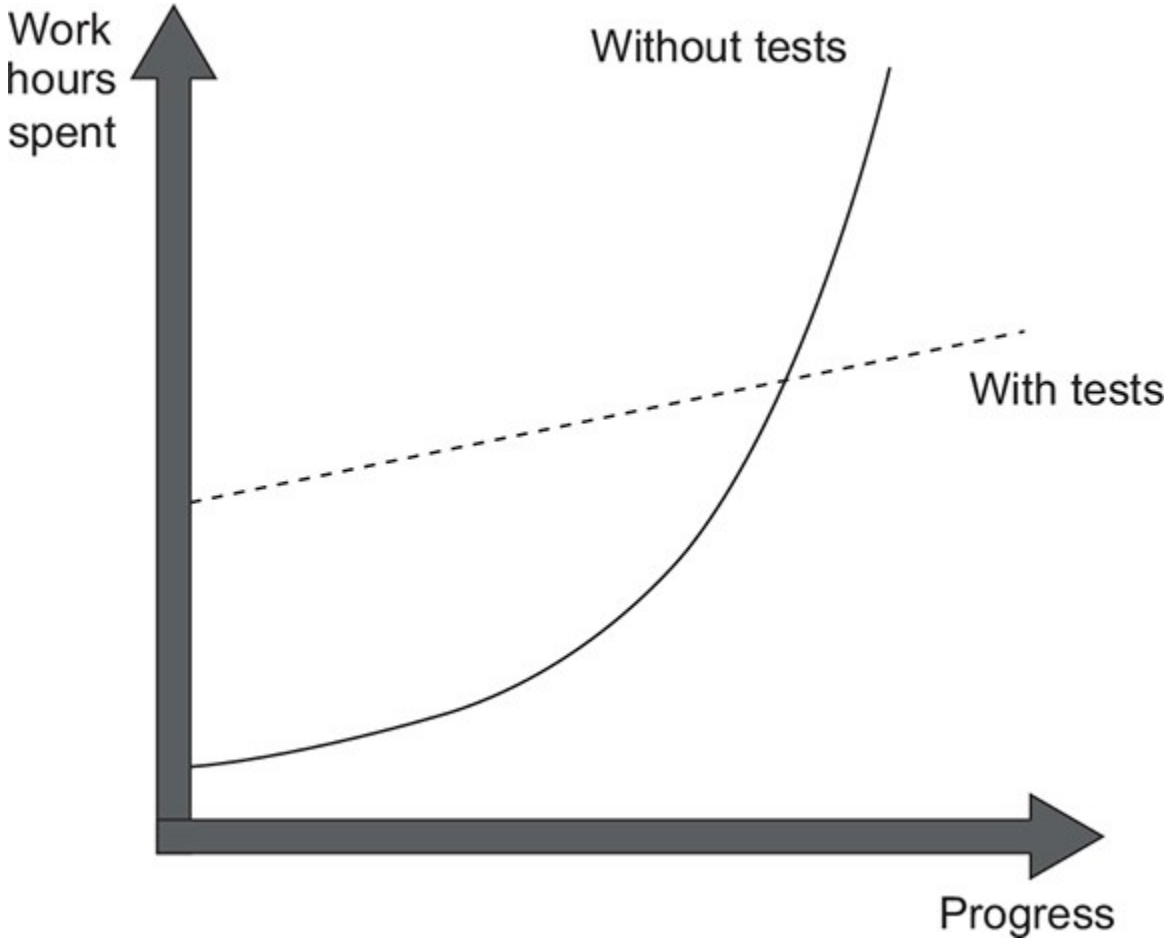
Teste unitário

Fatec
Rio Claro

O objetivo é permitir o crescimento sustentável do projeto de software. O termo "sustentável" é fundamental. É muito fácil desenvolver um projeto, especialmente quando se começa do zero. É muito mais difícil sustentar esse crescimento ao longo do tempo.

Teste unitário

Fatec
Rio Claro



KHORIKOV, Vladimir. Unit testing: principles, practices, and patterns. Shelter Island, NY: Manning Publications, 2020.

Teste unitário

Fatec
Rio Claro

Você começa rápido porque não há nada te atrapalhando. Nenhuma decisão arquitetônica ruim foi tomada ainda, e não há código existente com o qual se preocupar.

Com o passar do tempo, no entanto, você precisa dedicar cada vez mais horas para alcançar o mesmo progresso que demonstrou no início.

Com o tempo, a velocidade de desenvolvimento diminui significativamente, às vezes até o ponto em que você não consegue fazer nenhum progresso.

Cobertura de testes como métrica

Fatec
Rio Claro

Cobertura de testes (ou test coverage) mede quanto do seu código é executado pelos testes. Ou seja, quais linhas, condições e funções são de fato testadas.

$$\text{Code coverage (test coverage)} = \frac{\text{Lines of code executed}}{\text{Total number of lines}}$$

Alta cobertura não garante que não há bugs, mas ajuda a mostrar o que está ou não testado.

Fatec
Rio Claro

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br