

# Template de Apoio à Atividade de Teste de Software

Criado para o Exercício Final da UC 13 - Executar Teste e Implantação de Aplicativos Computacionais  
Turma 109 - Tecnico de informatica - Alunos: Debora Rafaelle e Victor Barros Roma

## 1. Cenário geral do rock\_paper\_scissors.py (o quê será testado)

O algoritmo é um jogo de pedra, papel e tesoura. Ele recebe a opção informada pelo usuário e a compara com a opção sorteada para o algoritmo, determinando se o usuário ganhou, perdeu ou empatou. A funcionalidade a ser testada será a entrada do usuário. Sera feito testes de

## 2. Estratégia(s) de Teste (como será testado)

- Será realizado um teste de funcionalidade dos tipos de entrada do usuário, sendo elas: letras maiúsculas, palavras completas, letras com números, duas opções e letras com caracteres especiais, com o objetivo de verificar quais são as entradas válidas e inválidas.
- Será gerado um grafo de fluxo de controle, onde serão realizados os testes dos nós e arcos acionados, com o objetivo de verificar todos os nós e arcos acessados. Para isso, será utilizado o **pycfg** para criar o grafo de controle de fluxo (CFG) e enumerar os nós, a fim de registrar o fluxo.
- Serão criados testes para verificar vitória, derrota e empate com a utilização do **pytest** e **mutpy**. Os testes de mutação serão gerados para as funções "play" e "is\_win", com a finalidade de verificar a eficiência dos testes criados, de modo que seus mutantes sejam variações mortas.

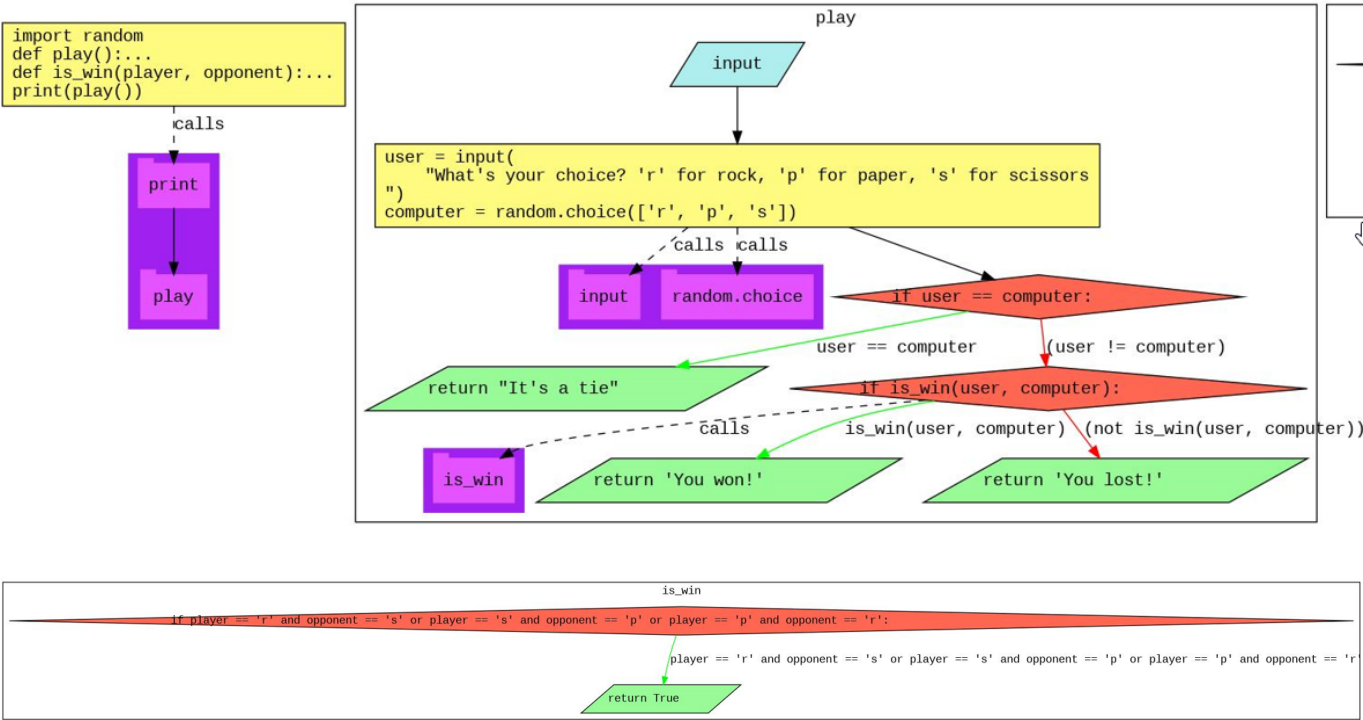
## 3. Projeto de Casos de Teste (como será testado)

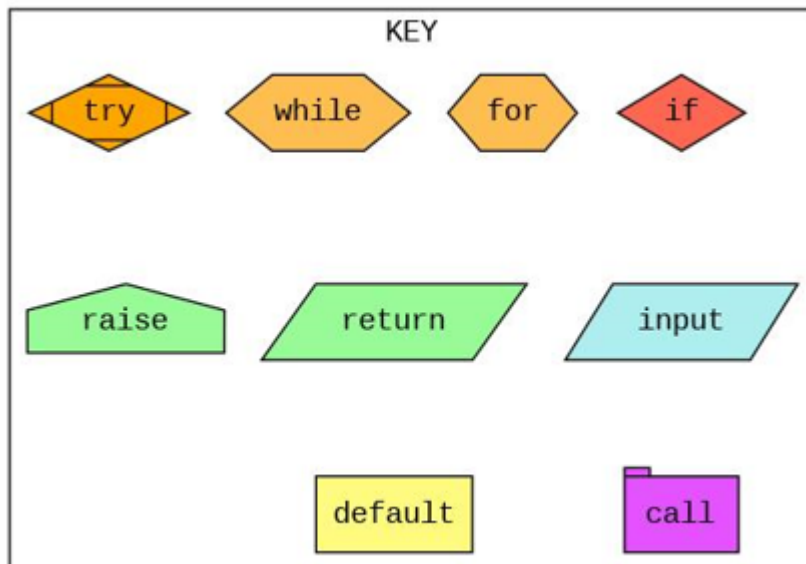
### - TESTE FUNCIONAL

ID	Módulo	Descrição	Roteiro	Resultado Esperado
1	Entrada do usuário com letras maiúsculas.	Testar a entrada com caracteres maiúsculos na escolha.	Ao executar o código, quando solicitado, responder "P".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.
2	Entrada do usuário com a palavra completa da opção desejada.	Testar a entrada com palavras completas da opção desejada pelo usuário.	Ao executar o código, quando solicitado, responder "ROCK".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.

ID	Módulo	Descrição	Roteiro	Resultado Esperado
3	Entrada do usuário com letras e números na opção desejada.	Testar a entrada com letras e números na opção desejada pelo usuário.	Ao executar o código, quando solicitado, responder "sc1ss0rs".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.
4	Entrada do usuário com duas opções desejadas.	Testar a entrada com duas opções desejadas pelo usuário.	Ao executar o código, quando solicitado, responder "p e".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.
5	Entrada do usuário com caracteres especiais na opção desejada.	Testar a entrada com caracteres especiais na opção desejada pelo usuário.	Ao executar o código, quando solicitado, responder "p@per#".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.
6	Entrada do usuário com a primeira letra minúscula da opção desejada.	Testar a entrada com a primeira letra minúscula na opção desejada pelo usuário.	Ao executar o código, quando solicitado, responder "r".	Ser informado se o usuário ganhou, perdeu ou empatou com a máquina.

- TESTE ESTRUTURAL DE NOS E ARCOS





A partir do CFG gerado acima pelo pycfg a cima será enumerado os nos e a partir disso serão testados quais nos e arcos são atingidos dependendo da sua entrada

#### - TESTE MUTACIONAIS

A partir dos testes criados para as funções "play" e "is\_win" será utilizado a biblioteca do **mutpy** com a utilização do **pytest** como runner

```
# Testes para a função is_win

# Testes para condições de vitória
def test_is_win_rock_beats_scissors():
    assert is_win('r', 's') == True # Pedra vence tesoura

def test_is_win_scissors_beats_paper():
    assert is_win('s', 'p') == True # Tesoura vence papel

def test_is_win_paper_beats_rock():
    assert is_win('p', 'r') == True # Papel vence pedra

# Testes para condições de derrota
def test_is_win_rock_loses_to_paper():
    assert is_win('r', 'p') != False # Pedra perde para papel

def test_is_win_scissors_loses_to_rock():
    assert is_win('s', 'r') != False # Tesoura perde para pedra

def test_is_win_paper_loses_to_scissors():
    assert is_win('p', 's') != False # Papel perde para tesoura

# Testes para a função play

# Teste de vitória
def test_play_win(monkeypatch: MonkeyPatch):
    monkeypatch.setattr('builtins.input', lambda _: 'r') # Simula a entrada do usuário como 'r'
    monkeypatch.setattr(random, 'choice', lambda _: 's') # Simula a escolha da máquina como 's'
    result = play() # Chama a função play()
    assert result == 'You won!' # Verifica se o resultado está correto

# Teste de derrota
def test_play_lose(monkeypatch: MonkeyPatch):
    monkeypatch.setattr('builtins.input', lambda _: 'r') # Simula a entrada do usuário como 'r'
    monkeypatch.setattr(random, 'choice', lambda _: 'p') # Simula a escolha da máquina como 'p'
    result = play() # Chama a função play()
    assert result == 'You lost!' # Verifica se o resultado está correto

# Teste de empate
def test_play_tie(monkeypatch: MonkeyPatch):
    monkeypatch.setattr('builtins.input', lambda _: 'r') # Simula a entrada do usuário como 'r'
    monkeypatch.setattr(random, 'choice', lambda _: 'r') # Simula a escolha da máquina como 'r'
    result = play() # Chama a função play()
    assert result == "It's a tie" # Verifica se o resultado está correto
```

4. Execução (quando e como será testado)

- TESTE FUNCIONAL

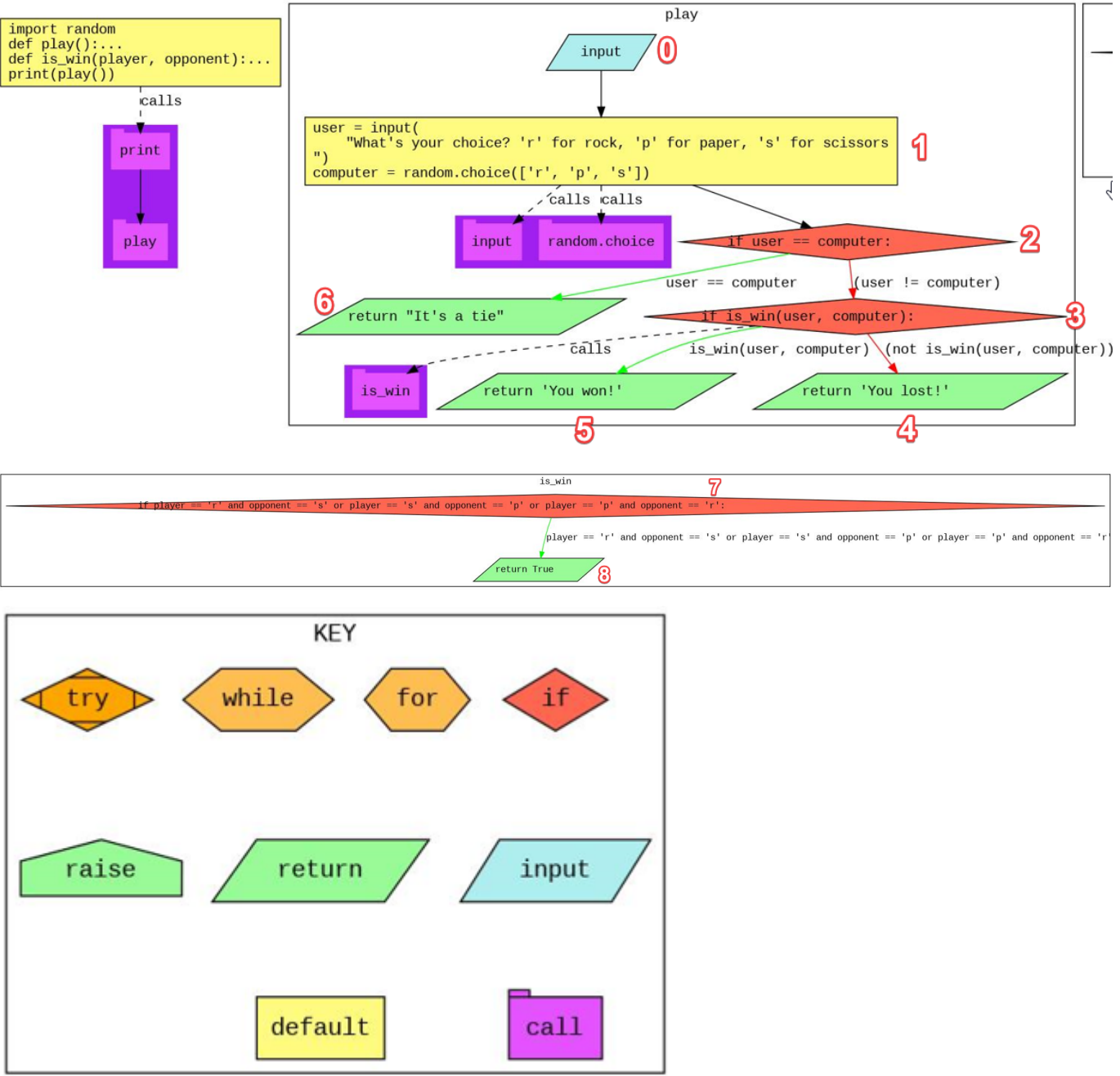
ID	Roteiro	Resultado Esperado	Resultado do Teste	Problema Encontrado
----	---------	--------------------	--------------------	---------------------

ID	Roteiro	Resultado Esperado	Resultado do Teste	Problema Encontrado
1	Ao executar o código, quando solicitado, responder "P".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário perdeu em vários testes.	Ao repetir o teste mais de uma vez, o resultado é sempre a derrota, pois não há um comparativo para a condição de derrota, ou seja, não é verificada a entrada do usuário e a opção da máquina.
2	Ao executar o código, quando solicitado, responder "ROCK".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário perdeu em vários testes.	Ao repetir o teste mais de uma vez, o resultado é sempre a derrota, pois não há um comparativo para a condição de derrota, ou seja, não é verificada a entrada do usuário e a opção da máquina.
3	Ao executar o código, quando solicitado, responder "sc1ss0rs".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário perdeu em vários testes.	Ao repetir o teste mais de uma vez, o resultado é sempre a derrota, pois não há um comparativo para a condição de derrota, ou seja, não é verificada a entrada do usuário e a opção da máquina.
4	Ao executar o código, quando solicitado, responder "p e".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário perdeu em vários testes.	Ao repetir o teste mais de uma vez, o resultado é sempre a derrota, pois não há um comparativo para a condição de derrota, ou seja, não é verificada a entrada do usuário e a opção da máquina.
5	Ao executar o código, quando solicitado, responder "p@per#".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário perdeu em vários testes.	Ao repetir o teste mais de uma vez, o resultado é sempre a derrota, pois não há um comparativo para a condição de derrota, ou seja, não é verificada a entrada do usuário e a opção da máquina.
6	Ao executar o código, quando solicitado, responder "r".	Informar se o usuário ganhou, perdeu ou empatou com a máquina.	O programa informou que o usuário ganhou, perdeu e empatou em testes diferentes.	Nenhum problema encontrado durante as repetições do teste.

Os testes foram realizados aplicando os valores desejados pelo roteiro do teste com o programa sendo executado no visual studio code

- TESTE ESTRUTURAL DE NOS E ARCOS

- CFG NUMERADO



- NOS

ENTRADA	SORTEADO COMPUTADOR	CAMINHOS DOS NOS
r (rock)	s (scissors)	0, 1, 2, 3, 7, 8, 3, 5
p (paper)	p (paper)	0, 1, 2, 6
s (scissors)	r (rock)	0, 1, 2, 3, 7, 3, 4

OBS: Todos os NOS foram ativados com os testes realizados

- ARCOS

ENTRADA	SORTEADO COMPUTADOR	ARCOS FEITOS
r (rock)	s (scissors)	0-1, 1-2, 2-3, 3-7, 7-8, 8-3, 3-5
p (paper)	p (paper)	0-1, 1-2, 2-6
s (scissors)	r (rock)	0-1, 1-2, 2-3, 3-7, 7-3, 3-4

**OBS: Todos os ARCOS foram ativados com os testes realizados**

Com o auxilio do CFG gerado pelo pycfg para cada simulação dos nos e arcos vendo os caminhos de acordo com o dado da entrada e sorteado do computador propostos pelo teste.

- TESTE MUTACIONAIS



Target

- rock\_paper\_scissors

Tests [9]

- test\_rock\_paper\_scissors [0.073 s]

Result summary

-  Score - 92.9%
-  Time - 0.9 s

Mutants [14]

-  killed - 13
-  survived - 1
-  incompetent - 0
-  timeout - 0

-Mutações geradas que foram mortas:  
3 de COI (conditional operator deletion)  
3 de LCR (logical connector replacement)  
7 de ROR (relational operator replacement)

Totais de 13 mutações foram mortas

-Mutações geradas que sobreviveram:  
1 de LCR (logical connector replacement)

Total de 1 mutação sobreviveu

## 5. Análise dos Resultados e Próximos Passos

Após a realização dos testes nas diferentes formas as conclusões de cada um foram:

### - TESTE FUNCIONAL

Ao analisar os resultados dos testes realizados vemos que o programa somente aceita um formato de entrada, deixando de fora outras formas de informar a opção desejada pelo usuário e tendo o problema de quando utilizada outras formas o programa sempre cai na opção de derrota mesmo que seja uma entrada que não é válida.

### - TESTE ESTRUTURAL DE NOS E ARCOS

A partir do uso do CFG realizando teste tendo como resultado: vitória, derrota e empate é possível checar que todos os NOS e ARCOS são passados.

### - TESTE MUTACIONAIS

Analisando os resultados das mutações geradas pelos conjuntos de teste temos um total de 14 mutações onde uma sobreviveu, ao analisar a mutação sobrevivente vimos que se trata de uma condição do programa que tem muitos termos o que possibilitou a mutação sobreviver.

### - Sugestão de mudanças no código

1. Uma forma de verificar se a entrada do usuário é válida ou inválida para necessidade do programa.
2. Outras formas de entradas permitidas para a necessidade do programa
3. A declaração das condições de derrota para o programa, atualmente ela acontece por as condições de vitória e empate não serem atendidas sendo assim qualquer variação de entradas que causam ou não vitória ou empate caem nessa condição que declara que o usuário perdeu.
4. Na função `is_win` esta uma única condição englobando todos os casos de vitória, o que acaba acontecendo uma mutação que fica viva, sugerimos a separação dessas condições.
5. Ao entrar na função `is_win` se nenhuma das condições de ganhar é comprovada não retorna como uma condição falsa, em um de nossos testes não executaram por não ter a possibilidade da condição como falsa.