Exercícios - Semana 09

O objetivo da atividade desta semana consiste em observar funções e descrever quais parâmetros poderiam ser utilizados para efetivamente testar tal implementação. Além disso, sua equipe poderá emitir um parecer sobre a implementação feita. Tem algum ponto de melhoria ?

Esta atividade poderá ser feito em dupla ou trio, **preferencialmente com pessoas de outras equipes da Napp**. Enviar a atividade no classroom e preencher o formulário. No formulário, você deve informar o nome de quem trabalhou com você nesta atividade.

Não é necessário implementar os testes com o pytest ou qualquer outra ferramenta.

Problema 1

Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ?

```
def juros_simples(capital, taxa=0.1, n_periodos=2):
    if capital <= 0:
        raise Exception('Capital precisa ser maior que zero.')
    if taxa < 0 or taxa > 1:
        raise Exception('Taxa precisa estar entre 0 e 1.')
    if n_periodos <= 0:
        raise Exception('Período precisa ser maior que zero.')
    return capital + capital * taxa * n_periodos
```

Problema 2

Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ?

```
def soma_numeros(*args):
    soma = 0
    for item in args:
        try:
            soma = soma + item
        except TypeError:
            raise TypeError('Incompatibilidade de tipos. Verificar parâmetros')
    return soma
```

Problema 3

Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ?

```
import json


def criar_json(**kwargs):
    return json.dumps(kwargs)
```

Problema 4
Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ?

```
def situacao_escolar(nota_final, ausencias=0):
    if ausencias < 0 or ausencias > 80:
        raise Exception('Ausências entre 0 e 80')
    if nota_final < 0 or nota_final > 10:
        raise Exception('Nota Final entre 0 e 10')
    if ausencias > 40:
        return 'Reprovado por falta'
    if nota_final < 5:
        return 'Reprovado por nota'
    if nota_final < 7:
        return 'Reprovado, em regime de recuperação'
    return 'Aprovado'
```

Problema 5
Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ? Quais melhorias/refatorações podem ser feitas ?

```
def vacina_ja(idade, **kwargs):
    profissao_com_prioridade = ['medico', 'enfermeiro']
    profissao_com_prioridade += ['medica', 'enfermeira']
    profissao_com_prioridade += ['auxiliar de enfermagem']
    profissao_com_prioridade += ['profissionais da saude']
    profissao = kwargs.get('profissao', '').lower()
    if profissao in profissao_com_prioridade:
        return 'Autorizado Vacinação'
    if idade >= 69:
        return 'Autorizado Vacinação'
    if profissao == 'professor' and idade > 47:
        return 'Autorizado Vacinação'
    return 'Não autorizado por enquanto'
```

Problema 6
Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ? Quais melhorias/refatorações podem ser feitas ?

```
lista_meses = ['janeiro', 'fevereiro', 'março', 'abril', 'maio']
lista_meses += ['junho', 'julho', 'agosto', 'setembro']
lista_meses += ['outubro', 'novembro', 'dezembro']

meses = dict(zip(
    list(range(1, 13)),
    lista_meses))


def data_por_extenso(data):
    data = data.split('/')
    try:
        return data[0] + ' de ' + meses[int(data[1])] + ' de ' + data[2]
    except KeyError:
        raise KeyError('Mês inválido')
```

Problema 7
Observe o código abaixo. Quais testes poderiam ser criados para validar a implementação abaixo ?

```
def n_medias(*notas, **kwnotas):
    media = 0
    if notas:
        media = sum(notas)/float(len(notas))
    elif kwnotas:
        media = sum(kwnotas.values())/float(len(kwnotas))
    return media
```

Problema 8

Observe as implementações 8A e 8B. Ambas realizam a mesma tarefa computacional. Qual das duas implementações é mais fácil para ser testada automaticamente? **Justifique sua resposta**

```
'''
8A: Cálculo de média simples.
'''


def calculo_media():
    numero_1 = float(input('Digite o primeiro número:  '))
    numero_2 = float(input('Digite o segundo número:  '))
    numero_3 = float(input('Digite o terceiro número:  '))
    media = (numero_1 + numero_2 + numero_3) / 3
    print('A média é: ' + str(media))


calculo_media()
```

```
'''
8B: Cálculo de média simples.
'''


def calculo_media(numero_1, numero_2, numero_3):
    return (numero_1 + numero_2 + numero_3) / 3


numero1 = float(input('Digite o primeiro número:  '))
numero2 = float(input('Digite o segundo número:  '))
numero3 = float(input('Digite o terceiro número:  '))

media = str(calculo_media(numero1, numero2, numero3))
print('A média é: ' + media)
```