Listas.py

```
# Criação da lista mesclada
lista_mesclada = [1, 2, 3, "Olá, Python", True, 12.6]
print("Conteúdo da lista:", lista_mesclada)

# Adicionando uma lista aninhada usando o método append
lista_mesclada.append(["Lista aninhada"])
print("Conteúdo da lista após append:", lista_mesclada)

# Inserindo o número 5 na posição 4
lista_mesclada.insert(4, 5)
print("Conteúdo da lista após insert:", lista_mesclada)

# Imprimindo o tamanho atual da lista
print("Tamanho da lista:", len(lista_mesclada))

# Removendo o item da posição 1
lista_mesclada.pop(1)
print("Conteúdo da lista após remover o item da posição 1:", lista_mesclada)

# Criando uma nova lista com os primeiros 4 elementos
nova_lista_mesclada = lista_mesclada[:4]
print("Conteúdo da nova lista mesclada:", nova_lista_mesclada)
```

Tuplas.py

```
# Criação da tupla
primeira_tupla = (1, 2, 3, 4, "Olá, tupla")

# Imprimindo o conteúdo da tupla
print("Conteúdo da tupla:", primeira_tupla)

# Imprimindo o índice do elemento 4
indice_do_quatro = primeira_tupla.index(4)
print("Índice do elemento 4:", indice_do_quatro)

# Verificando se a tupla contém o elemento 3
contem_tres = 3 in primeira_tupla
print("A tupla contém o elemento 3?", contem_tres)

# Verificando se a tupla contém o elemento 33
contem_trinta_tres = 33 in primeira_tupla
print("A tupla contém o elemento 33?", contem_trinta_tres)
```

Sets.py

```
# Criação do set inicial
set_inicial = {11, 12, 13, 14}
print("Conteúdo do set inicial:", set_inicial)

# Adicionando o elemento 15 ao set
set_inicial.add(15)
print("Conteúdo do set após adicionar 15:", set_inicial)
```

```
# Atualizando o set com novos elementos
set_inicial.update([1, 2, 3, 4, 5])
print("Conteúdo do set após a atualização:", set_inicial)

# Removendo o elemento 13 do set
set_inicial.discard(13)
print("Conteúdo do set após remover o 13:", set_inicial)

# Criação de um novo set com o método set()
novo_set = set([20, 21, 23, 1, 2])
print("Conteúdo do novo set:", novo_set)

# União dos dois sets
uniao_sets = set_inicial.union(novo_set)
print("União dos dois sets:", uniao_sets)

# Interseção dos dois sets
intersecao_sets = set_inicial.intersection(novo_set)
print("Interseção dos dois sets:", intersecao_sets)

# Diferença entre os dois sets
diferenca_sets = set_inicial.difference(novo_set)
print("Diferença entre os dois sets:", diferenca_sets)

# Diferença simétrica entre os dois sets
diferenca_simetrica_sets = set_inicial.symmetric_difference(novo_set)
print("Diferença simétrica entre os dois sets:", diferenca_simetrica_sets)
```

Dicionários.py

```
# Criação do dicionário meu_dicionario
meu_dicionario = {
   "codigo_1": "Python",
   "codigo_2": "Java",
   "codigo_3": "PHP"
}

# Imprimindo o conteúdo do dicionário
print("Conteúdo do dicionário:", meu_dicionario)

# Imprimindo o tipo de dados do dicionário
print("Tipo de dados do dicionário:", type(meu_dicionario))

# Utilizando o método get para imprimir o valor da chave "linguagem"
linguagem = meu_dicionario.get("codigo_1")
print("Valor da chave 'codigo_1':", linguagem)

# Imprimindo o tamanho do dicionário
print("Tamanho do dicionário:", len(meu_dicionario))

# Criação de um novo dicionário aninhado chamado dicionario_frutas
dicionario_frutas = {
   1: {"nome": "limão", "tipo": "ácida"},
   2: {"nome": "laranja", "tipo": "ácida"},
   3: {"nome": "manga", "tipo": "semiácida"},
   4: {"nome": "maçã", "tipo": "semiácida"},
```

```
    5: {"nome": "banana", "tipo": "doce"},
    6: {"nome": "mamão", "tipo": "doce"}
}

# Imprimindo o valor das chaves "nome" e "tipo" da chave 1
print("Chave 1 - Nome:", dicionario_frutas[1]["nome"], "- Tipo:", dicionario_frutas[1]["tipo"])

# Imprimindo o valor das chaves "nome" e "tipo" da chave 2
print("Chave 2 - Nome:", dicionario_frutas[2]["nome"], "- Tipo:", dicionario_frutas[2]["tipo"])

# Iterando no dicionário dicionario_frutas e imprimindo os valores das chaves "nome" e "tipo"
for chave, valor in dicionario_frutas.items():
    print(f"Chave {chave} - Nome: {valor['nome']} - Tipo: {valor['tipo']}")
```

Dicionários2.py

```
# Criação do dicionário inicial
meu_dicionario = {
    1: {'nome': 'Maria', 'idade': 26, 'nacionalidade': 'brasileira'}
}

# Utilizando o método update para adicionar novos elementos
meu_dicionario.update({
    2: {'nome': 'João', 'idade': 30, 'nacionalidade': 'português'},
    3: {'nome': 'Ana', 'idade': 22, 'nacionalidade': 'espanhola'}
})
```

```
# Imprimindo o dicionário atualizado
print("Dicionário atualizado:", meu_dicionario)

# Criando uma cópia do dicionário
copia_dicionario = meu_dicionario.copy()

# Removendo um elemento com pop e imprimindo o conteúdo
elemento_removido = meu_dicionario.pop(2)
print("Elemento removido (chave 2):", elemento_removido)
print("Dicionário após remoção com pop:", meu_dicionario)

# Removendo o último elemento com popitem e imprimindo o conteúdo
elemento_removido_final = meu_dicionario.popitem()
print("Último elemento removido com popitem:", elemento_removido_final)
print("Dicionário após remoção com popitem:", meu_dicionario)

# Removendo todos os elementos dos dicionários
meu_dicionario.clear()
copia_dicionario.clear()
print("Dicionário original após clear:", meu_dicionario)
print("Cópia do dicionário após clear:", copia_dicionario)

# Usando o método fromKeys para criar um novo dicionário
chaves = ['a', 'b', 'c']
novo_dicionario = dict.fromkeys(chaves, 'valor_padrao')

# Imprimindo o conteúdo do novo dicionário usando o método items
```

```
print("Conteúdo do novo dicionário (items):", novo_dicionario.items())

# Imprimindo apenas as chaves do novo dicionário usando o método keys
print("Chaves do novo dicionário:", novo_dicionario.keys())

# Imprimindo apenas os valores do novo dicionário usando o método values
print("Valores do novo dicionário:", novo_dicionario.values())
```

Operacoes.py

```
def calcular_media(notas):
    """
    Calcula a média das notas recebidas.
    :param notas: Lista com as notas dos 4 bimestres.
    :return: Média das notas.
    """
    return sum(notas) / len(notas)

def verificar_reprovacao(media):
    """
    Verifica se o aluno foi reprovado com base na média.
    :param media: Média das notas do aluno.
    :return: True se a média for inferior a 6 (reprovado), False caso contrário.
    """
    return media < 6

def alunos_reprovados(dados_alunos, matriculas_reprovados):
    """
    Retorna a lista de alunos reprovados.
```

```
    :param dados_alunos: Dicionário com os dados dos alunos.
    :param matriculas_reprovados: Lista com as matrículas dos alunos reprovados.
    :return: Mensagem com os alunos reprovados.
    """
    for matricula in matriculas_reprovados:
        aluno = dados_alunos[matricula]
        media = calcular_media(aluno["notas"])
        print(f'Aluno Reprovado: {aluno["nome"]} – Matrícula: {matricula} – Média Final: {media:.2f}')
```

Main.py

```
from operacoes import calcular_media, verificar_reprovacao, alunos_reprovados

# Dados dos alunos
dados_alunos = {
    26: {"nome": "Maria", "notas": [8, 7, 5, 9]},
    101: {"nome": "Ana", "notas": [9, 9, 8, 9]},
    13: {"nome": "João", "notas": [6, 5, 5, 5]},
    37: {"nome": "Ágatha", "notas": [8, 6, 7.5, 9]},
    72: {"nome": "Joaquim", "notas": [6, 5.5, 5, 7]},
    5: {"nome": "Félix", "notas": [10, 8, 8, 8]}
}

# Lista de matrículas dos alunos reprovados
matriculas_reprovados = []

# Verificação da média e aprovação/reprovação de cada aluno
for matricula, dados in dados_alunos.items():
```

```
    media = calcular_media(dados["notas"])
    if verificar_reprovacao(media):
        matriculas_reprovados.append(matricula)

# Exibição dos alunos reprovados
alunos_reprovados(dados_alunos, matriculas_reprovados)
```