Nama : Fadhil Zulfikar (11231024)
Muhammad Yunus (11231066)

```
def hitungTotal(x,y):
 t=0
 for i in range(len(x)):
  t=t+x[i]*y[i]
 return t
```

QA (Quality Assurance)

1. Identifikasi minimal 3 pelanggaran coding convention pada kode diatas.
2. Lakukan refactoring kode agar lebih sesuai dengan standar QA (gunakan nama variabel yang jelas, komentar singkat, dan format rapi).
3. Buatlah QA checklist sederhana (minimal 5 poin) untuk kode diatas.

Jawab :

1. Pelanggaran pada kode ;
   - Nama variabel terlalu singkat: x, y, t, dan i tidak deskriptif,
   - Fungsi tidak memiliki docstring/komentar penjelas,
   - Nama fungsi sebaiknya menggunakan snake_case (hitung_total) bukan camelCase (hitungTotal).
   - Logika pada line 4

2. Refactoring

tes.py > ...

```python
def hitung_total (harga, total) :
    """menghitung harga total belanjaan"""
    totalHarga = 0
    for i in range (len(harga)) :
        totalHarga += harga[i] * total[i] #perkalian tiap elemen
        return totalHarga
```

3.

No.	Item	Yes	No	N/A	Comment
1.	Apakah nama variabel jelas dan deskriptif?				
2.	Apakah terdapat docstring/komentar penjelas pada fugsi utama?				
3.	Apakah format penulisan kode sudah rapi?				
4.	Apakah tidak ada kode yang duplikat?				
5.	Apakah kode sudah di riview?				

QC (Quality Control)
Studi Kasus

Aplikasi Kasir menerima input daftar barang dan jumlah. Kode berikut digunakan untuk menghitung total harga:

```
items = [
    {"name": "Pensil", "price": 2000, "qty": 2},
    {"name": "Buku", "price": 5000, "qty": 1},
    {"name": "Pulpen", "price": 3000, "qty": 2}
]

def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total
```

Pertanyaan dan Jawaban

1. Buatlah minimal 4 test case dalam bentuk tabel (Skenario, input, Hasil yang diharapkan, Hasil Aktual, Status).

Jawaban :

Skenario	Input	Hasil yang diharapkan	Hasil Aktual	Status
Hitung total 2 Barang	Nama Barang : “Buku”, Harga : 5000, Jumlah : 2	Total = 10000	?	Pass/Fail
Uji Quantity Float	Nama Barang: Pulpen, Harga : 3000, Jumlah : 5,3	Ditolak (invalid)	?	Pass/Fail
Uji Quantity = 0	Nama Barang : :”Pensil”, Harga : 2000, Jumlah : 0	Total = 0	?	Pass/Fail
Uji Price negatif	Nama Barang : “Buku”, Harga : -5000, Jumlah : 2	Ditolak (invalid)	?	Pass/Fail
Uji Quantity Negatif	Nama Barang : “Pensil”, Harga : 2000, jumlah : -2	Ditolak(invalid)	?	Pass/Fail
Uji price string	Nama Barang : “Buku”, Harga : “dua ribu”, Jumlah : 1	DItolak(invalid)	?	Pass/Fail
Uji Quantity String	Nama Barang : “Pulpen”, Harga : 2000, jumlah : “Tiga”	Ditolak(invalid)	?	Pass/Fail

2. Jalankan kode di atas dengan test case buatanmu. Cata hasil aktualnya

Jawaban :

A. Hitung total 2 barang

```python
# Daftar barang
items = [
    {"name": "Buku", "price": 5000, "qty": 2},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
10000
```

B. Uji Quantity Float

```python
# Daftar barang
items = [
    {"name": "Pulpen", "price": 3000, "qty": 5.3},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
15900.0
```

C. Uji Quantity = 0

```python
# Daftar barang
items = [
    {"name": "Pensil", "price": 2000, "qty": 0},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
0
```

D. Uji Price Negatif

```python
# Daftar barang
items = [
    {"name": "Buku", "price": -5000, "qty": 2},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
-10000
```

E. Uji Quantity Negatif

```python
# Daftar barang
items = [
    {"name": "Pensil", "price": 2000, "qty": -2},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
-4000
```

F. Uji Price String

```python
# Daftar barang
items = [
    {"name": "Buku", "price": "duaribu", "qty": 1},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most recent
/tmp/ipython-input-1289813028.py in <cell line: 0>()
     11     return total
     12
```

G. Uji Quantity String

```python
# Daftar barang
items = [
    {"name": "Pulpen", "price": "2000", "qty": "Tiga"},
]

# Fungsi untuk menghitung total belanja
def calculate_total(items):
    total = 0
    for item in items:
        total += item["price"] * item["qty"]
    return total

calculate_total(items)
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most recent ca
/tmp/ipython-input-1258018854.py in <cell line: 0>()
     11     return total
```

Skenario	Input	Hasil yang diharapkan	Hasil Aktual	Status
Hitung total 2 Barang	Nama Barang : “Buku”, Harga : 5000, Jumlah : 2	Total = 10000	10000	Pass/Fail
Uji Quantity Float	Nama Barang: Pulpen, Harga : 3000, Jumlah : 5,3	Ditolak (invalid)	15900.0	Pass/Fail
Uji Quantity = 0	Nama Barang : :”Pensil”, Harga : 2000, Jumlah : 0	Total = 0	0	Pass/Fail
Uji Price negatif	Nama Barang : “Buku”, Harga : -5000, Jumlah : 2	Ditolak (invalid)	-10000	Pass/Fail
Uji Quantity Negatif	Nama Barang : “Pensil”, Harga : 2000, jumlah : -2	Ditolak(invalid)	-4000	Pass/Fail
Uji price string	Nama Barang : “Buku”, Harga : “dua ribu”, Jumlah : 1	DItolak(invalid)	Error (type)	Pass/Fail
Uji Quantity String	Nama Barang : “Pulpen”, Harga : 2000, jumlah : “Tiga”	Ditolak(invalid)	Error (type)	Pass/Fail

3. Tentukan statu pass/fail pada tiap case.

Skenario	Input	Hasil yang diharapkan	Hasil Aktual	Status
Hitung total 2 Barang	Nama Barang : “Buku”, Harga : 5000, Jumlah : 2	Total = 10000	10000	Pass
Uji Quantity Float	Nama Barang: Pulpen, Harga : 3000, Jumlah : 5,3	Ditolak (invalid)	15900.0	Fail
Uji Quantity = 0	Nama Barang : :”Pensil”, Harga : 2000, Jumlah : 0	Total = 0	0	Pass
Uji Price negatif	Nama Barang : “Buku”, Harga : -5000, Jumlah : 2	Ditolak (invalid)	-10000	Fail
Uji	Nama Barang :	Ditolak(invalid)	-4000	Fail

Quantity Negatif	“Pensil”, Harga : 2000, jumlah : -2			
Uji price string	Nama Barang : “Buku”, Harga : “dua ribu”, Jumlah : 1	Ditolak(invalid)	Error (type)	Pass
Uji Quantity String	Nama Barang : “Pulpen”, Harga : 2000, jumlah : “Tiga”	Ditolak(invalid)	Error (type)	Pass

4. Berikan kesimpulan apakah fungsi calculate_total() sudah memenuhi requirement.
   Jawaban :

   Berdasarkan hasil uji coba pada kode dengan fungsi calculate_total(), Didapatkan hasil bahwa fungsi calculate_total() belum memenuhi requirement yang diharapkan. Hal ini terjadi karena pada tabel hasil masih terdapat status uji yang tidak lulus/gagal, seperti pada uji quantity float, uji price negatif, dan uji quantity negatif.