```
"""
# A list of coins allowed
ACCEPTABLE_COINS = [200, 100, 25, 10, 5]

# A mapping of products and their prices
PRODUCT_PRICES = {
    'drink': 275,
    'chips': 225,
    'candy': 315
}

def insert_coin(coin, inserted_coins):
    """
    Accepts a coin and appends it to inserted_coins list.
    """
    if coin not in ACCEPTABLE_COINS:
        raise ValueError
    inserted_coins.append(coin)


def return_change(balance):
    """
    Returns balance in coins.
    """
    change = []
    balance -= balance % 5

    while balance > 0:
        for coin in ACCEPTABLE_COINS:
            if balance % coin == 0:
                change.append(coin)
                balance -= coin
                break

    return sorted(change, reverse=True)


def buy_product(product, balance):
    """
    Debits the balance using the product's price. A sufficient
    balance must be provided to complete the purchase.
    """
    if product not in PRODUCT_PRICES:
        raise ValueError
    price = PRODUCT_PRICES[product]

    if balance < price:
        raise InsufficientFunds

    return balance - price


class InsufficientFunds(Exception):
    """
    Exception used to indicate that there isn't
    enough money to make a purchase.
    """
```

Basic outline

This i can put it in class & do

Vending Machine

Program Requirements

1. Insert coins

- Write a function called `insert_coin` which takes two arguments: one called `coin` and another called `inserted_coins`.
- The `coin` parameter will accept the values any of the following values: 5, 10, 25, 100, 200. These values represents cents. Any other value should raise a `ValueError` exception.
- The "inserted_coins" parameter is a list
- When the function insert_coin() is called, the "coin" value is appended to the insert_coins parameter.

2. Buy a product

- Write a function called buy_product which takes two arguments: one called "product" and the other is called "balance".
- The product argument may be one of the following values: "drink", "candy", "chips". Any other value should raise a `ValueError` exception.
- `balance` is an integer representing the current balance of funds available
- When the `buy_product` function is called, the price of the chosen product is compared to the balance.
- If the balance of the inserted coins is less than the cost of the product, a custom exception called `InsufficientFunds` should be raised.
- If the balance equals or exceeds the cost of the product, then the function will return the balance minus the cost of the purchased product.
- The product costs are:
  - Drink: $2.75
  - Chips: $2.25
  - Candy: $3.15

3. Return change

- Write a function called return_change which takes a single argument called "balance".
- If the balance is greater than zero, return a list of coins (can be any combination of 5, 10, 25, 100, 200) which will sum up to the balance.
- If for whatever reason the balance is not a multiple of 5, then the sum of coins returned should be rounded down to the nearest multiple of 5, and not exceed the balance.
- The coins returned should be the largest first, then the smallest.