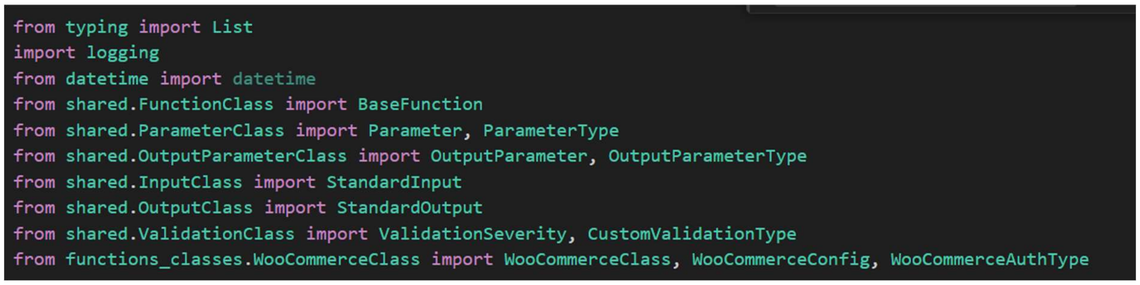
Structure how each CRUD operation should look like (Example: Coupon)

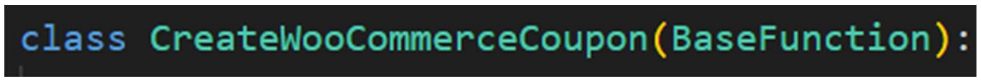
CREATE

1. First all imports:



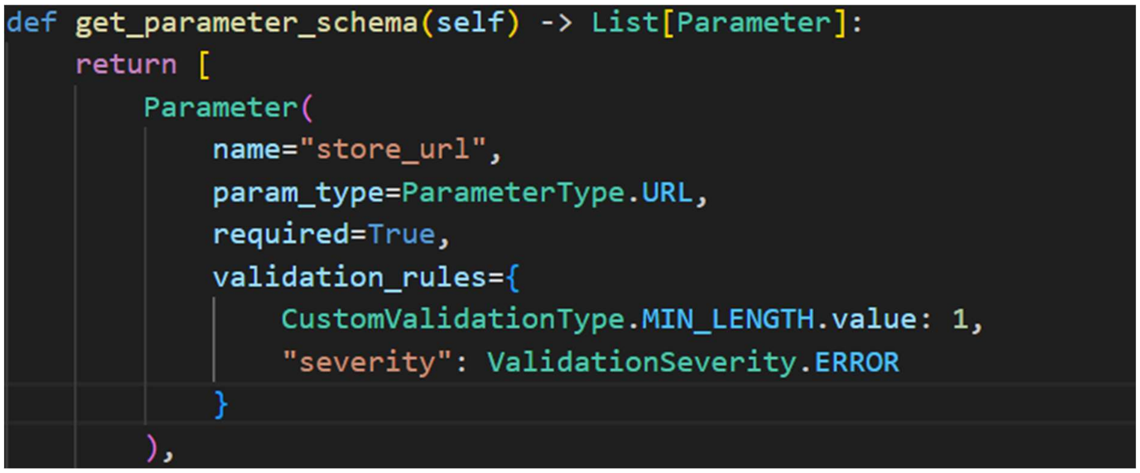
```python
from typing import List
import logging
from datetime import datetime
from shared.FunctionClass import BaseFunction
from shared.ParameterClass import Parameter, ParameterType
from shared.OutputParameterClass import OutputParameter, OutputParameterType
from shared.InputClass import StandardInput
from shared.OutputClass import StandardOutput
from shared.ValidationClass import ValidationSeverity, CustomValidationType
from functions_classes.WooCommerceClass import WooCommerceClass, WooCommerceConfig, WooCommerceAuthType
```

2. Class name should be Create{APPLICATION}{OBJECT}



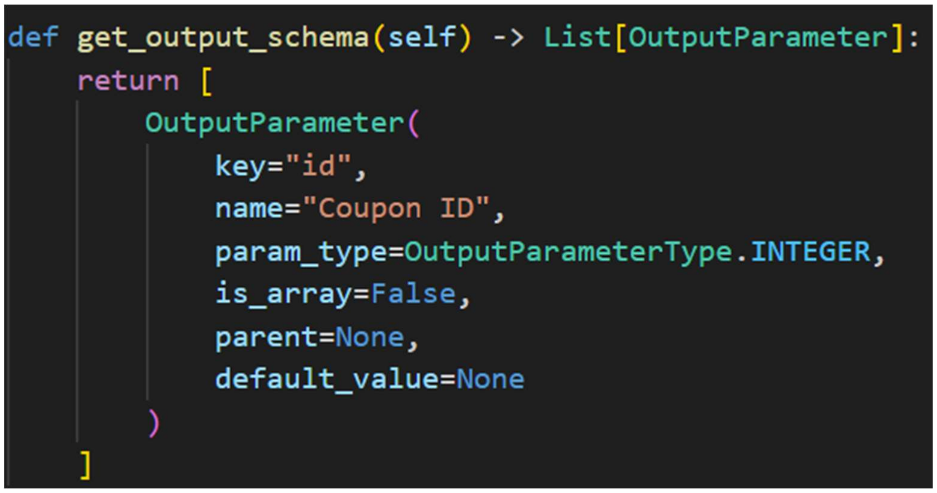
```python
class CreateWooCommerceCoupon(BaseFunction):
```

3. Generate the array of input variables (required should always be on the top)
   a. Make sure that it includes all optional values which can be used to create an object



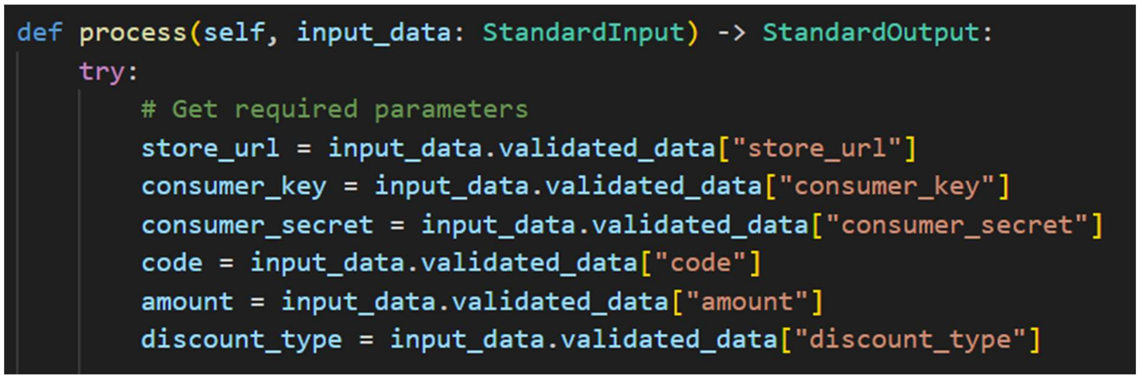
```python
def get_parameter_schema(self) -> List[Parameter]:
    return [
        Parameter(
            name="store_url",
            param_type=ParameterType.URL,
            required=True,
            validation_rules={
                CustomValidationType.MIN_LENGTH.value: 1,
                "severity": ValidationSeverity.ERROR
            }
        ),
```

4. Generate an array of output variables (for create only id is necessary, if applicable also a way to show it link the permalink)



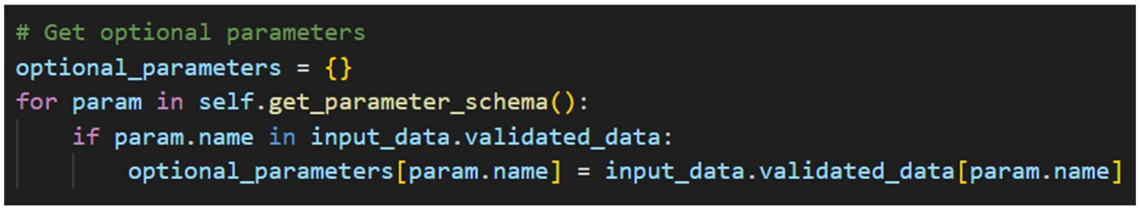
```python
def get_output_schema(self) -> List[OutputParameter]:
    return [
        OutputParameter(
            key="id",
            name="Coupon ID",
            param_type=OutputParameterType.INTEGER,
            is_array=False,
            parent=None,
            default_value=None
        )
    ]
```

5. The process function should have the implementation of the function itself
    a. Always start with getting the required parameters



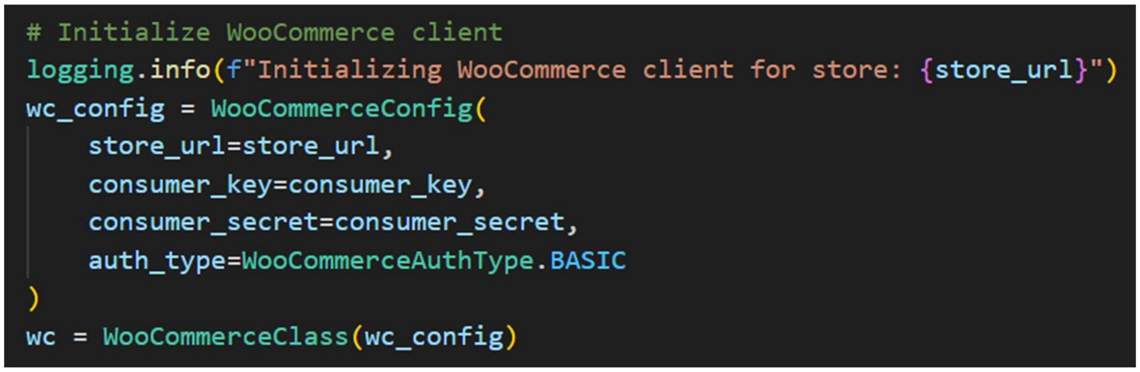
```python
def process(self, input_data: StandardInput) -> StandardOutput:
    try:
        # Get required parameters
        store_url = input_data.validated_data["store_url"]
        consumer_key = input_data.validated_data["consumer_key"]
        consumer_secret = input_data.validated_data["consumer_secret"]
        code = input_data.validated_data["code"]
        amount = input_data.validated_data["amount"]
        discount_type = input_data.validated_data["discount_type"]
```

    b. Then dynamically receive all the optional value



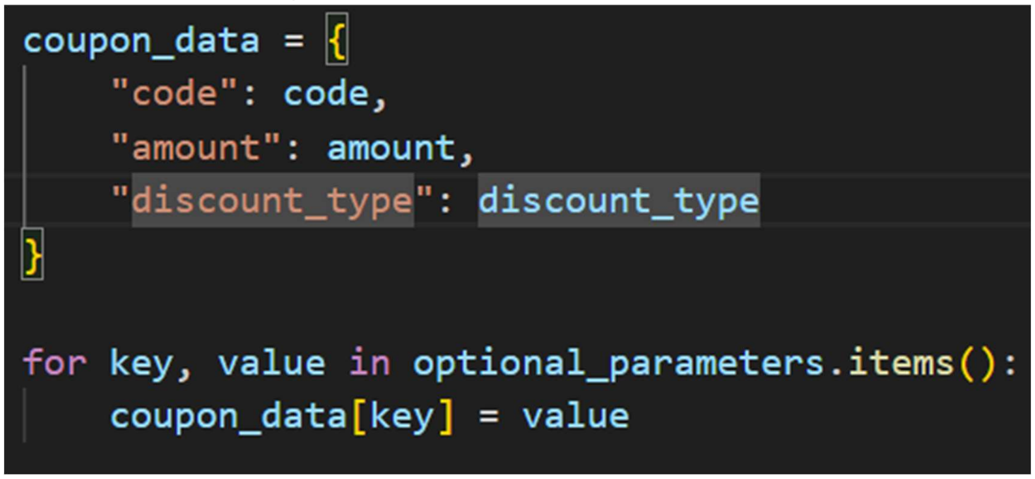
```python
# Get optional parameters
optional_parameters = {}
for param in self.get_parameter_schema():
    if param.name in input_data.validated_data:
        optional_parameters[param.name] = input_data.validated_data[param.name]
```

    c. Then initialize the class



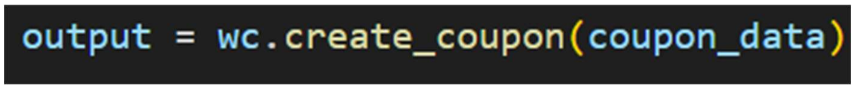
```python
# Initialize WooCommerce client
logging.info(f"Initializing WooCommerce client for store: {store_url}")
wc_config = WooCommerceConfig(
    store_url=store_url,
    consumer_key=consumer_key,
    consumer_secret=consumer_secret,
    auth_type=WooCommerceAuthType.BASIC
)
wc = WooCommerceClass(wc_config)
```

d. Format the data correctly for the function call



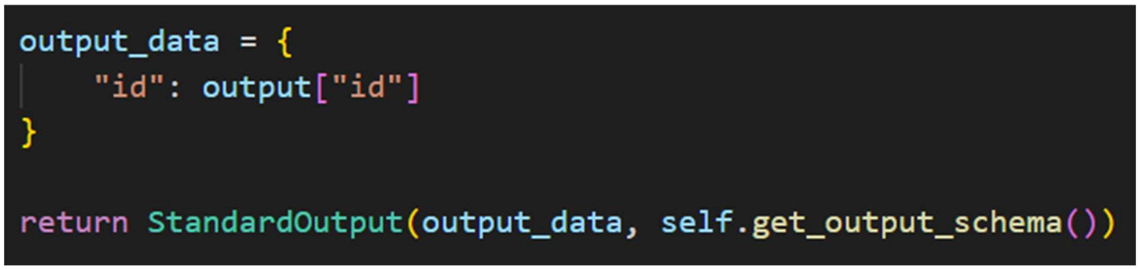
```
coupon_data = {
    "code": code,
    "amount": amount,
    "discount_type": discount_type
}

for key, value in optional_parameters.items():
    coupon_data[key] = value
```

e. Then make the call



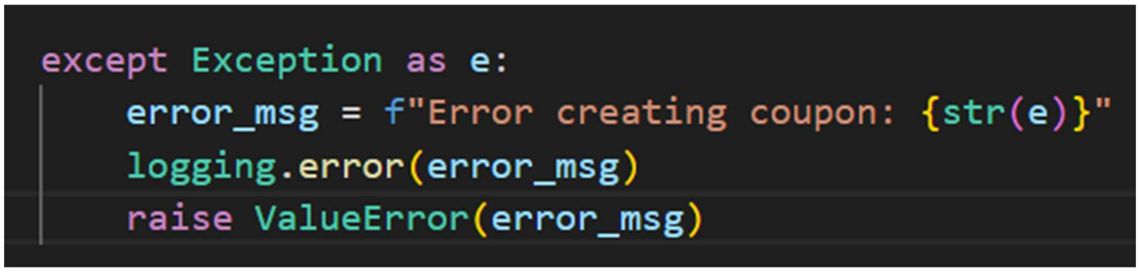
```
output = wc.create_coupon(coupon_data)
```

f. Return the data as is described in the output variables



```
output_data = {
    "id": output["id"]
}

return StandardOutput(output_data, self.get_output_schema())
```

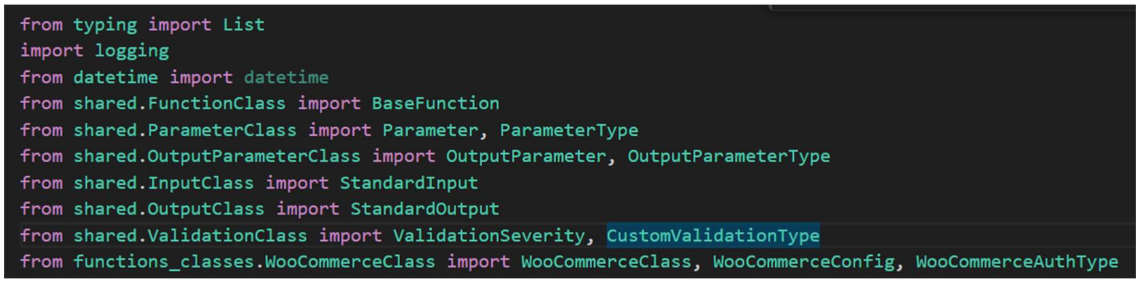
g. Make sure to catch all exceptions



```
except Exception as e:
    error_msg = f"Error creating coupon: {str(e)}"
    logging.error(error_msg)
    raise ValueError(error_msg)
```

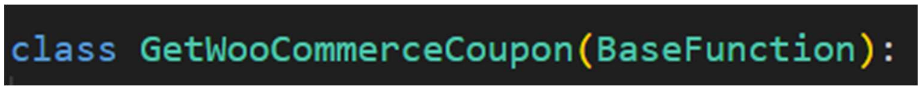
READ (Single)

1. First all imports



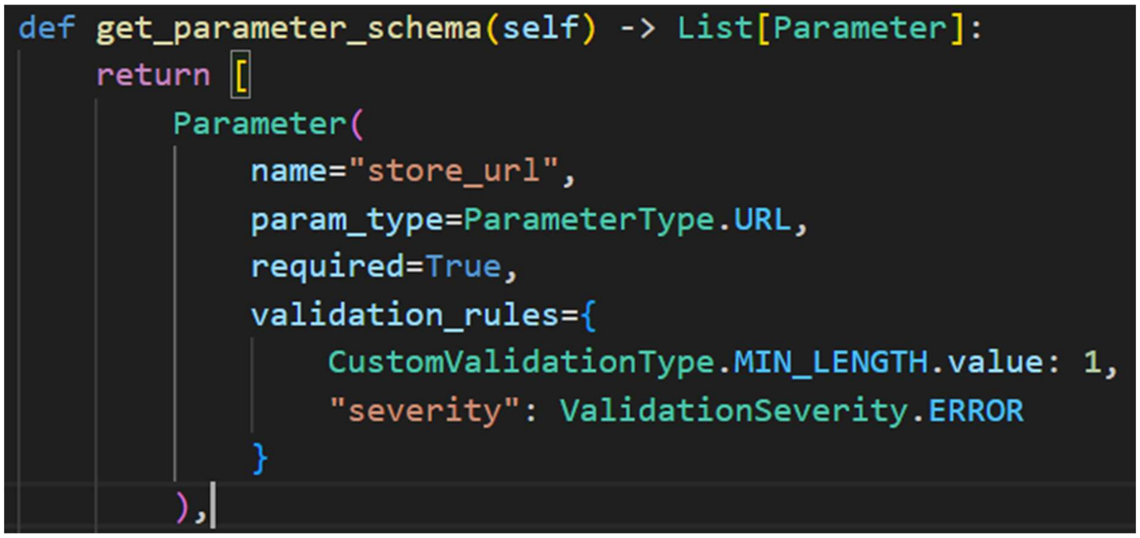
```python
from typing import List
import logging
from datetime import datetime
from shared.FunctionClass import BaseFunction
from shared.ParameterClass import Parameter, ParameterType
from shared.OutputParameterClass import OutputParameter, OutputParameterType
from shared.InputClass import StandardInput
from shared.OutputClass import StandardOutput
from shared.ValidationClass import ValidationSeverity, CustomValidationType
from functions_classes.WooCommerceClass import WooCommerceClass, WooCommerceConfig, WooCommerceAuthType
```

2. Class name should be Get{APPLICATION}{OBJECT}



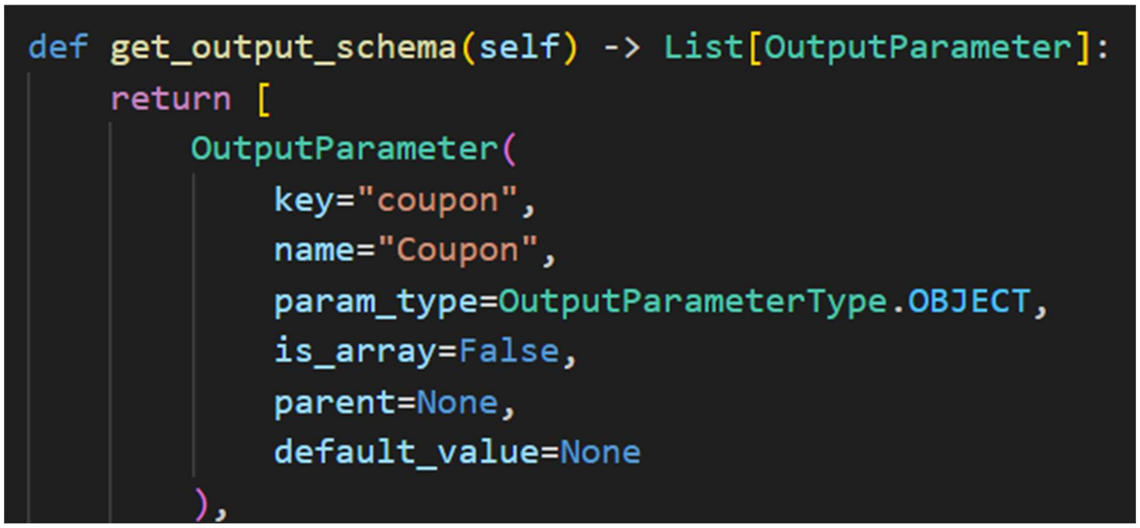
```python
class GetWooCommerceCoupon(BaseFunction):
```

3. Generate the array of input variables (required should always be on the top)
   a. For getting the object besides the authentication parameters only 1 identifier should be enough (like id or slug, etc)



```python
def get_parameter_schema(self) -> List[Parameter]:
    return [
        Parameter(
            name="store_url",
            param_type=ParameterType.URL,
            required=True,
            validation_rules={
                CustomValidationType.MIN_LENGTH.value: 1,
                "severity": ValidationSeverity.ERROR
            }
        ),
```

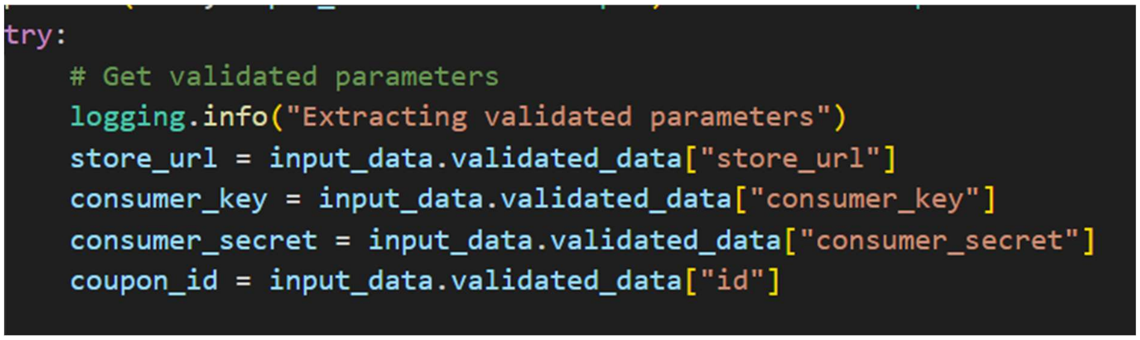
4. For Read (single) the whole object needs to be returned properly structured



```python
def get_output_schema(self) -> List[OutputParameter]:
    return [
        OutputParameter(
            key="coupon",
            name="Coupon",
            param_type=OutputParameterType.OBJECT,
            is_array=False,
            parent=None,
            default_value=None
        ),
```

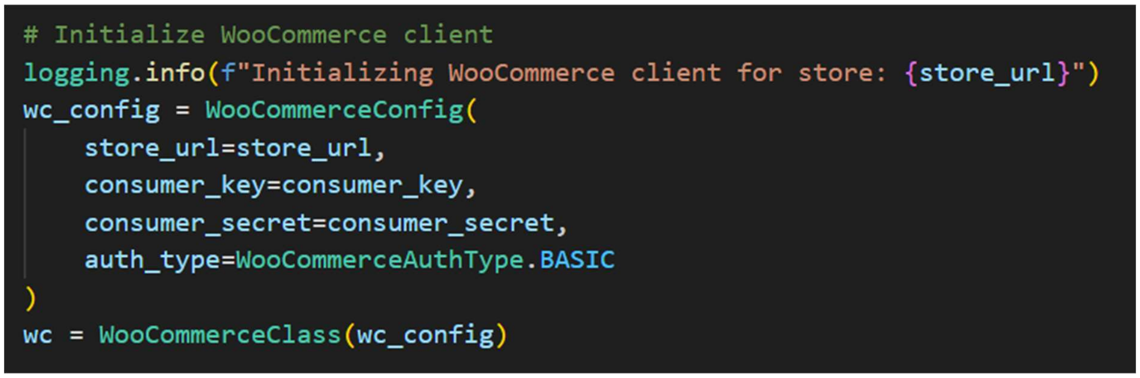
5. The process function should have the implementation of the function itself

a. Always start with getting the required parameters



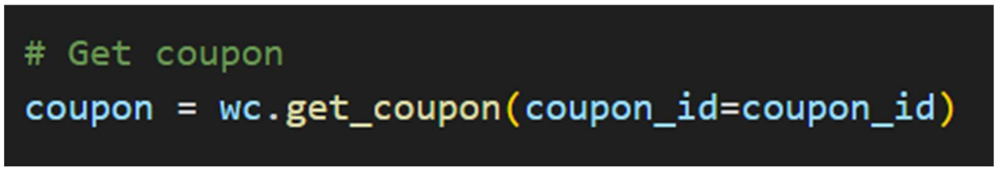
```python
try:
    # Get validated parameters
    logging.info("Extracting validated parameters")
    store_url = input_data.validated_data["store_url"]
    consumer_key = input_data.validated_data["consumer_key"]
    consumer_secret = input_data.validated_data["consumer_secret"]
    coupon_id = input_data.validated_data["id"]
```

b. Then initialize the class



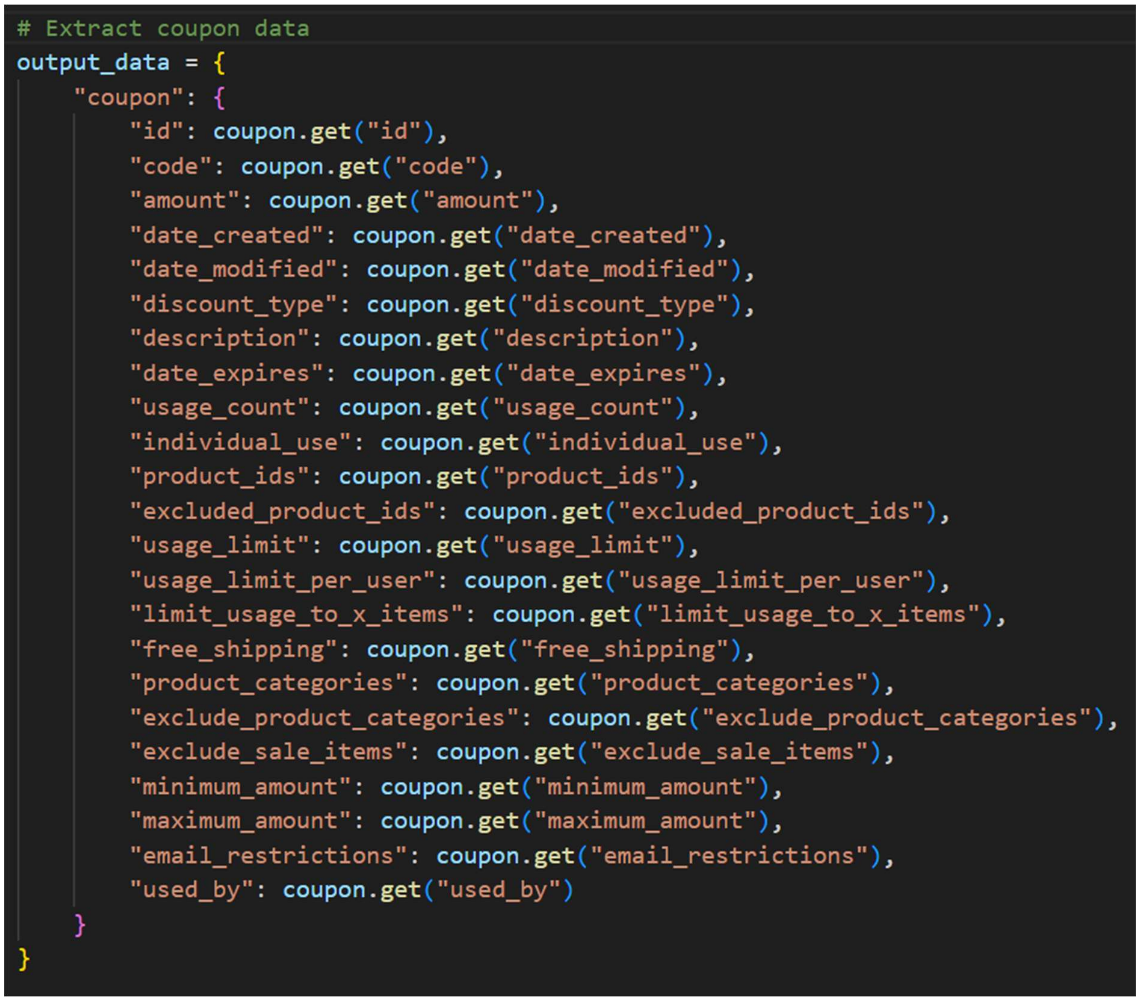
```python
# Initialize WooCommerce client
logging.info(f"Initializing WooCommerce client for store: {store_url}")
wc_config = WooCommerceConfig(
    store_url=store_url,
    consumer_key=consumer_key,
    consumer_secret=consumer_secret,
    auth_type=WooCommerceAuthType.BASIC
)
wc = WooCommerceClass(wc_config)
```

c. Then use the function from the class



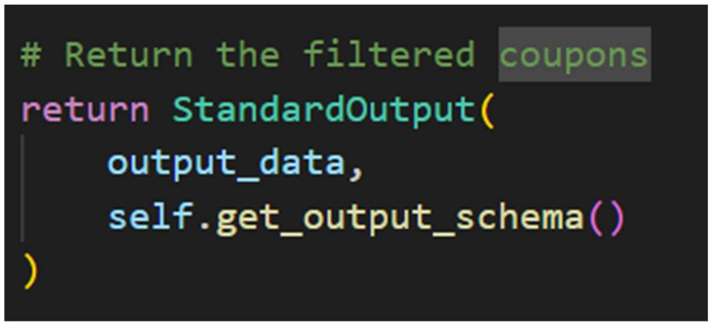
```python
# Get coupon
coupon = wc.get_coupon(coupon_id=coupon_id)
```

d. Format the object correctly (make sure that arrays are appended correctly to the object if arrays are present)



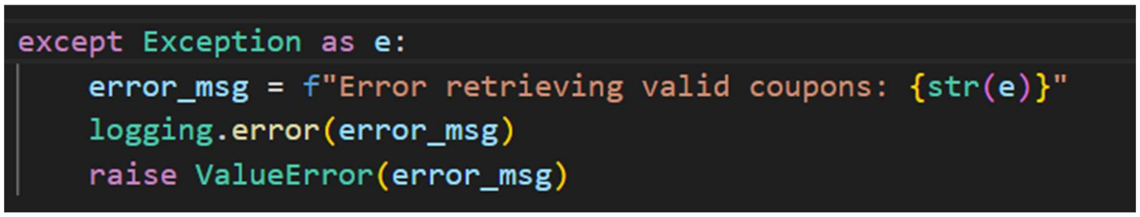
```python
# Extract coupon data
output_data = {
    "coupon": {
        "id": coupon.get("id"),
        "code": coupon.get("code"),
        "amount": coupon.get("amount"),
        "date_created": coupon.get("date_created"),
        "date_modified": coupon.get("date_modified"),
        "discount_type": coupon.get("discount_type"),
        "description": coupon.get("description"),
        "date_expires": coupon.get("date_expires"),
        "usage_count": coupon.get("usage_count"),
        "individual_use": coupon.get("individual_use"),
        "product_ids": coupon.get("product_ids"),
        "excluded_product_ids": coupon.get("excluded_product_ids"),
        "usage_limit": coupon.get("usage_limit"),
        "usage_limit_per_user": coupon.get("usage_limit_per_user"),
        "limit_usage_to_x_items": coupon.get("limit_usage_to_x_items"),
        "free_shipping": coupon.get("free_shipping"),
        "product_categories": coupon.get("product_categories"),
        "exclude_product_categories": coupon.get("exclude_product_categories"),
        "exclude_sale_items": coupon.get("exclude_sale_items"),
        "minimum_amount": coupon.get("minimum_amount"),
        "maximum_amount": coupon.get("maximum_amount"),
        "email_restrictions": coupon.get("email_restrictions"),
        "used_by": coupon.get("used_by")
    }
}
```

e. Return the data



```python
# Return the filtered coupons
return StandardOutput(
    output_data,
    self.get_output_schema()
)
```

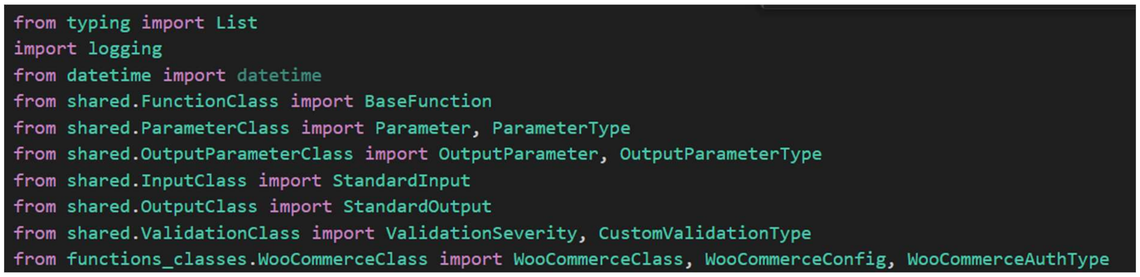
f. Make sure that all exceptions are catched



```python
except Exception as e:
    error_msg = f"Error retrieving valid coupons: {str(e)}"
    logging.error(error_msg)
    raise ValueError(error_msg)
```

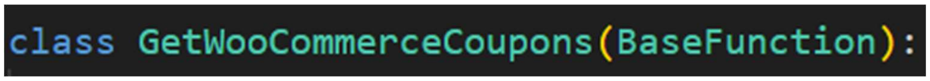
READ (Multiple)

1. First all imports



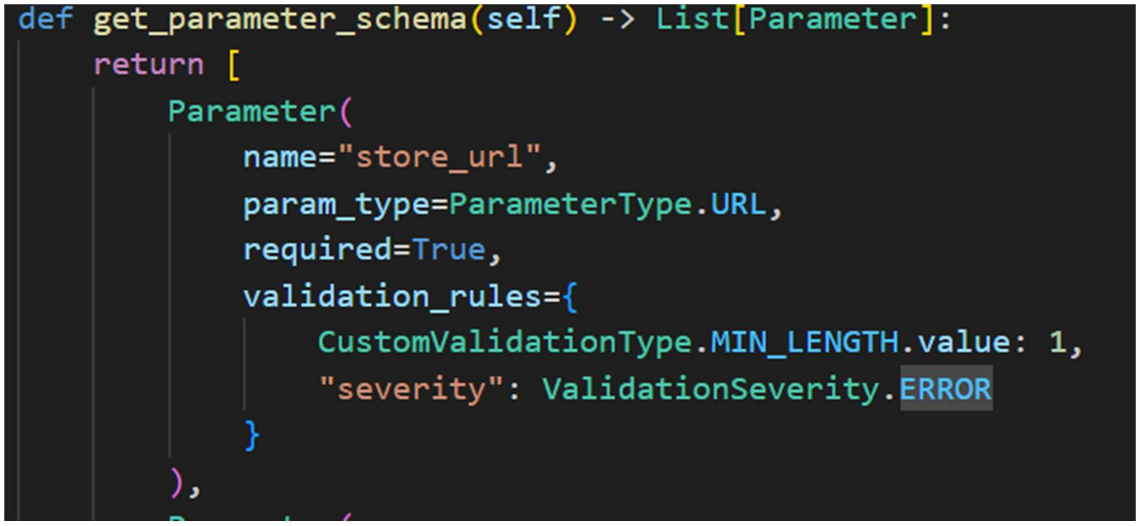
```
from typing import List
import logging
from datetime import datetime
from shared.FunctionClass import BaseFunction
from shared.ParameterClass import Parameter, ParameterType
from shared.OutputParameterClass import OutputParameter, OutputParameterType
from shared.InputClass import StandardInput
from shared.OutputClass import StandardOutput
from shared.ValidationClass import ValidationSeverity, CustomValidationType
from functions_classes.WooCommerceClass import WooCommerceClass, WooCommerceConfig, WooCommerceAuthType
```

2. Class name should be Get{APPLICATION}{OBJECTS}



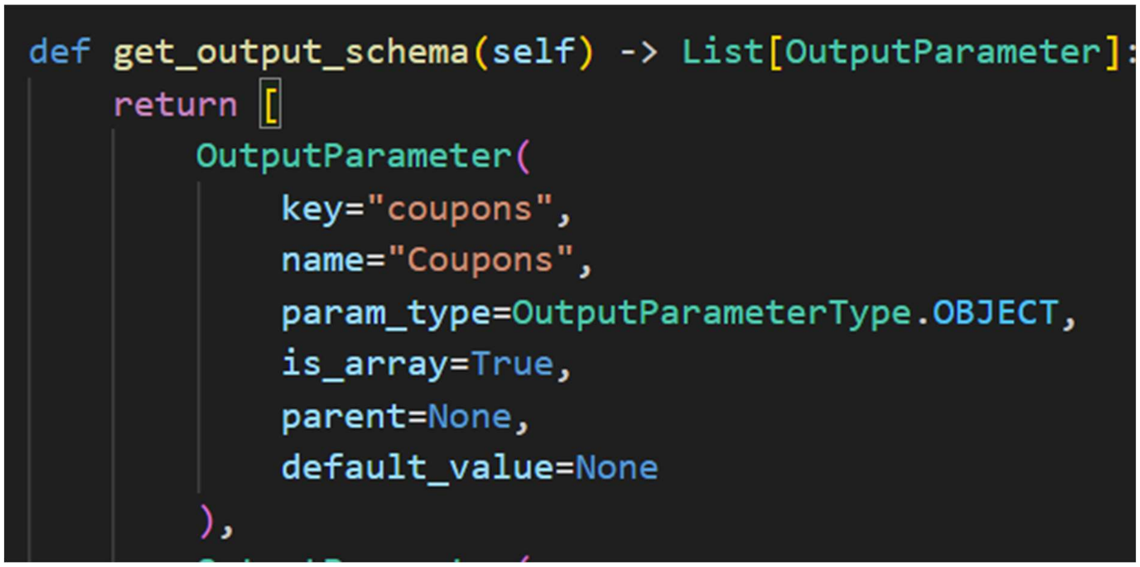
```
class GetWooCommerceCoupons(BaseFunction):
```

3. Generate the array of input variables (required should always be on the top)
   a. For getting the objects have the authentication parameters and the filtering options which are provided by the class



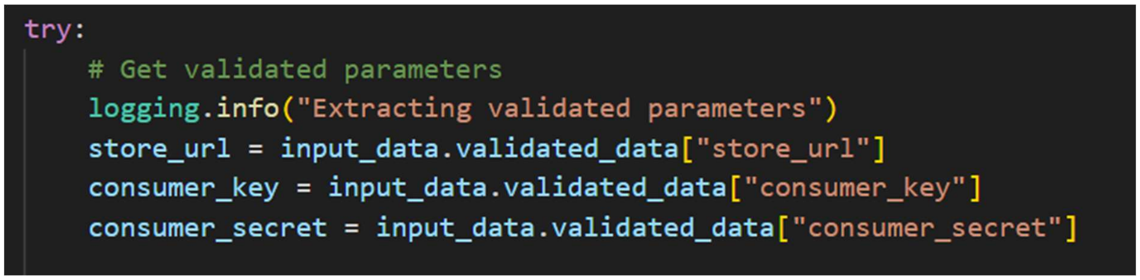
```
def get_parameter_schema(self) -> List[Parameter]:
    return [
        Parameter(
            name="store_url",
            param_type=ParameterType.URL,
            required=True,
            validation_rules={
                CustomValidationType.MIN_LENGTH.value: 1,
                "severity": ValidationSeverity.ERROR
            }
        ),
```

4. For Read (multiple) the whole object needs to be returned properly structured if possible, if not then the identifier + other data which is able to be returned



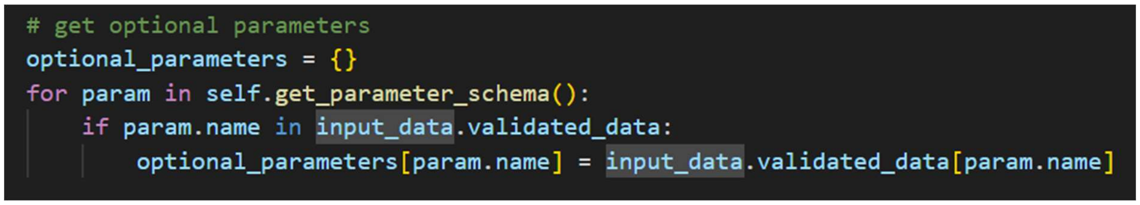
```
def get_output_schema(self) -> List[OutputParameter]:
    return [
        OutputParameter(
            key="coupons",
            name="Coupons",
            param_type=OutputParameterType.OBJECT,
            is_array=True,
            parent=None,
            default_value=None
        ),
```

5. The process function should have the implementation of the function itself
    a. Always start with getting the required parameters



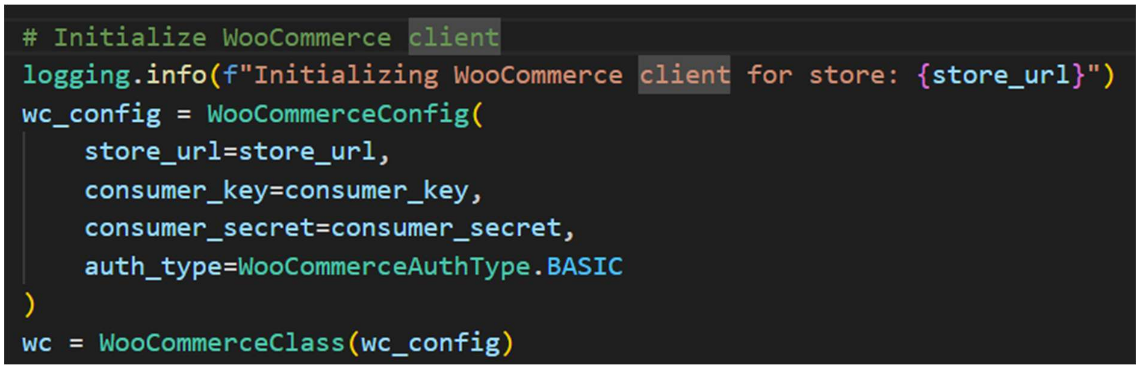
```
try:
    # Get validated parameters
    logging.info("Extracting validated parameters")
    store_url = input_data.validated_data["store_url"]
    consumer_key = input_data.validated_data["consumer_key"]
    consumer_secret = input_data.validated_data["consumer_secret"]
```

    b. Then get all the optional parameters



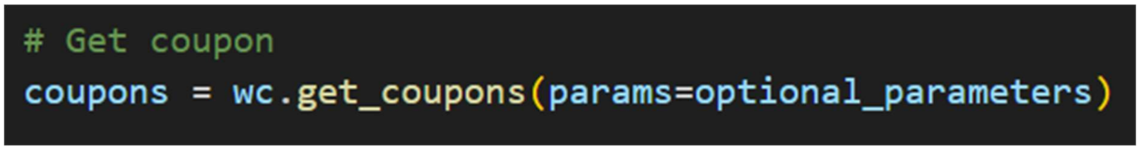
```
# get optional parameters
optional_parameters = {}
for param in self.get_parameter_schema():
    if param.name in input_data.validated_data:
        optional_parameters[param.name] = input_data.validated_data[param.name]
```

    c. Then initialize the class



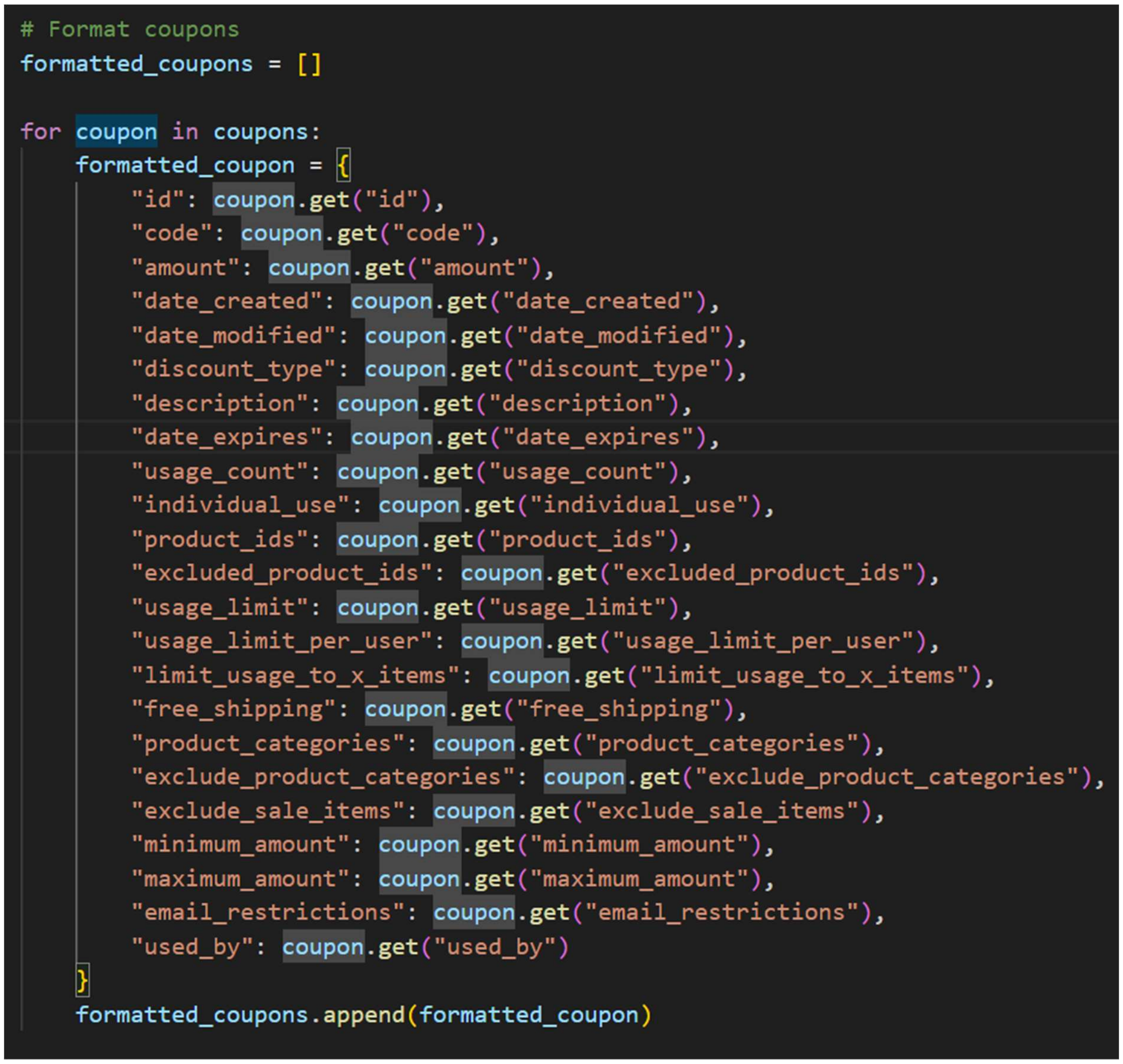
```
# Initialize WooCommerce client
logging.info(f"Initializing WooCommerce client for store: {store_url}")
wc_config = WooCommerceConfig(
    store_url=store_url,
    consumer_key=consumer_key,
    consumer_secret=consumer_secret,
    auth_type=WooCommerceAuthType.BASIC
)
wc = WooCommerceClass(wc_config)
```

    d. Then use the function from the class



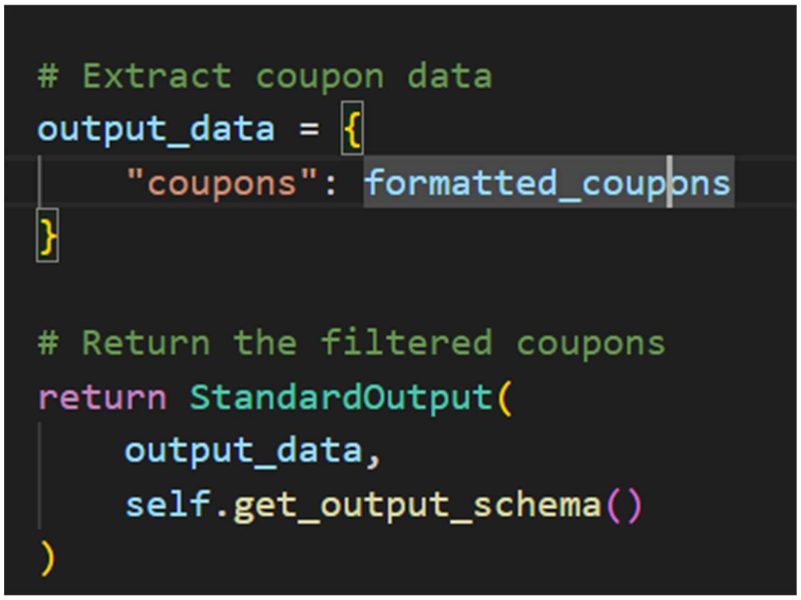
```
# Get coupon
coupons = wc.get_coupons(params=optional_parameters)
```

e. Format the object correctly in an array (make sure that arrays are appended correctly to the object if arrays are present inside the object)



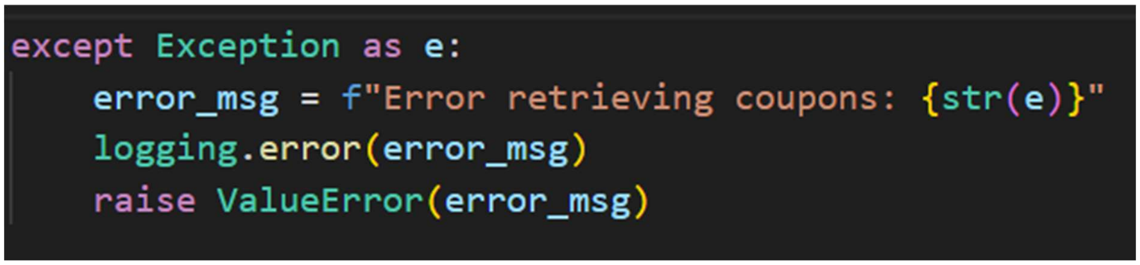
```python
# Format coupons
formatted_coupons = []

for coupon in coupons:
    formatted_coupon = {
        "id": coupon.get("id"),
        "code": coupon.get("code"),
        "amount": coupon.get("amount"),
        "date_created": coupon.get("date_created"),
        "date_modified": coupon.get("date_modified"),
        "discount_type": coupon.get("discount_type"),
        "description": coupon.get("description"),
        "date_expires": coupon.get("date_expires"),
        "usage_count": coupon.get("usage_count"),
        "individual_use": coupon.get("individual_use"),
        "product_ids": coupon.get("product_ids"),
        "excluded_product_ids": coupon.get("excluded_product_ids"),
        "usage_limit": coupon.get("usage_limit"),
        "usage_limit_per_user": coupon.get("usage_limit_per_user"),
        "limit_usage_to_x_items": coupon.get("limit_usage_to_x_items"),
        "free_shipping": coupon.get("free_shipping"),
        "product_categories": coupon.get("product_categories"),
        "exclude_product_categories": coupon.get("exclude_product_categories"),
        "exclude_sale_items": coupon.get("exclude_sale_items"),
        "minimum_amount": coupon.get("minimum_amount"),
        "maximum_amount": coupon.get("maximum_amount"),
        "email_restrictions": coupon.get("email_restrictions"),
        "used_by": coupon.get("used_by")
    }
    formatted_coupons.append(formatted_coupon)
```

f. Return the data



```python
# Extract coupon data
output_data = {
    "coupons": formatted_coupons
}

# Return the filtered coupons
return StandardOutput(
    output_data,
    self.get_output_schema()
)
```

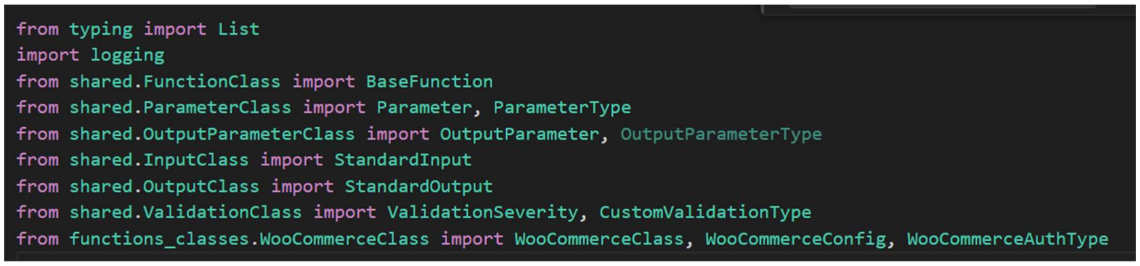
g. Make sure that all exceptions are catched



```python
except Exception as e:
    error_msg = f"Error retrieving coupons: {str(e)}"
    logging.error(error_msg)
    raise ValueError(error_msg)
```

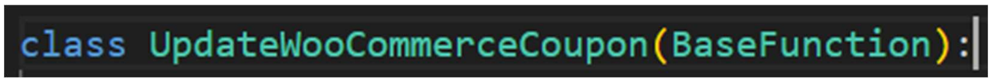
UPDATE

1. First all imports



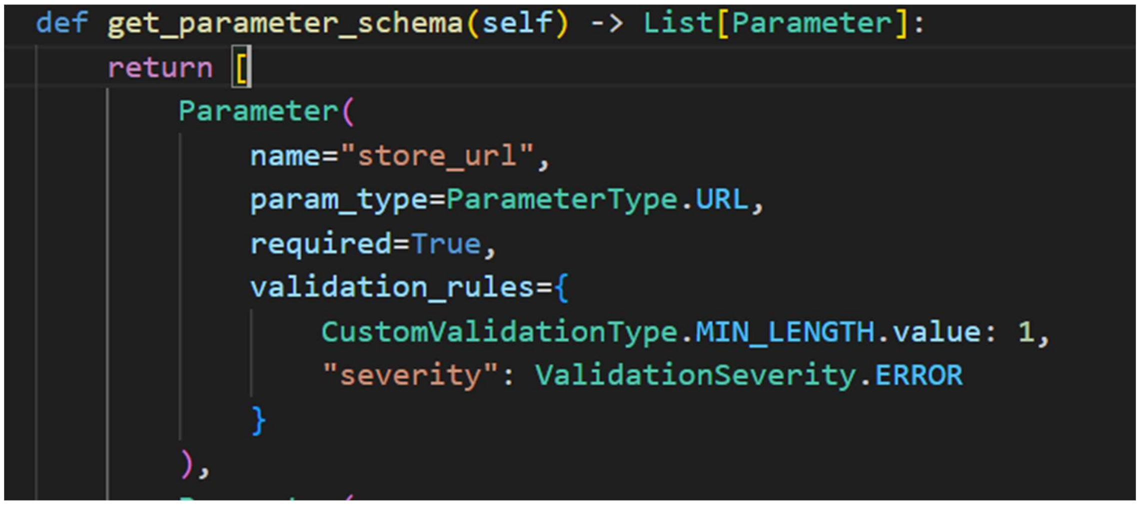
```
from typing import List
import logging
from shared.FunctionClass import BaseFunction
from shared.ParameterClass import Parameter, ParameterType
from shared.OutputParameterClass import OutputParameter, OutputParameterType
from shared.InputClass import StandardInput
from shared.OutputClass import StandardOutput
from shared.ValidationClass import ValidationSeverity, CustomValidationType
from functions_classes.WooCommerceClass import WooCommerceClass, WooCommerceConfig, WooCommerceAuthType
```

2. Class name should be Update{APPLICATION}{OBJECT}



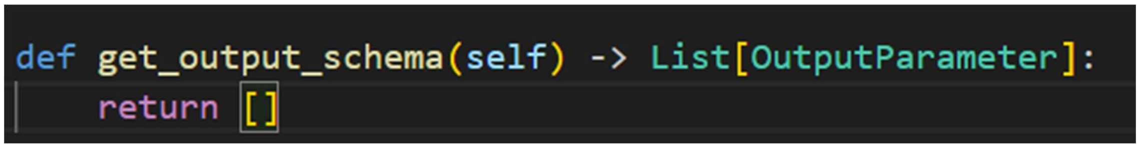
```
class UpdateWooCommerceCoupon(BaseFunction):
```

3. Generate the array of input variables (required should always be on the top)
   a. For getting the object besides the authentication parameters only an identifier should be present and all fields which can be updated



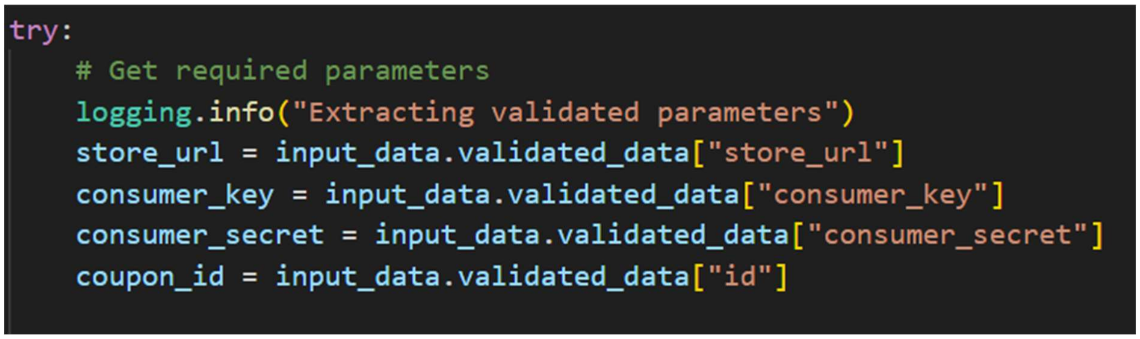
```
def get_parameter_schema(self) -> List[Parameter]:
    return [
        Parameter(
            name="store_url",
            param_type=ParameterType.URL,
            required=True,
            validation_rules={
                CustomValidationType.MIN_LENGTH.value: 1,
                "severity": ValidationSeverity.ERROR
            }
        ),
```

4. For update nothing needs to be returned



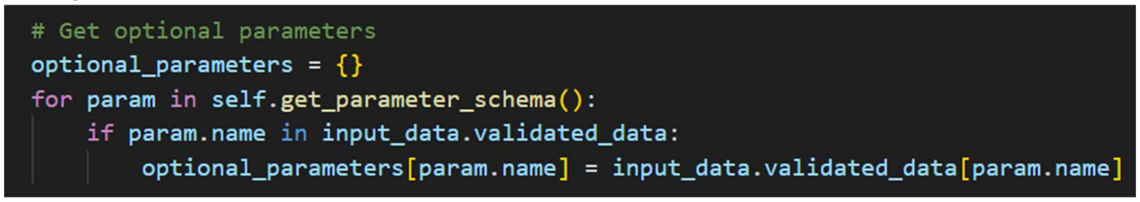
```
def get_output_schema(self) -> List[OutputParameter]:
    return []
```

5. The process function should have the implementation of the function itself
   a. Always start with getting the required parameters



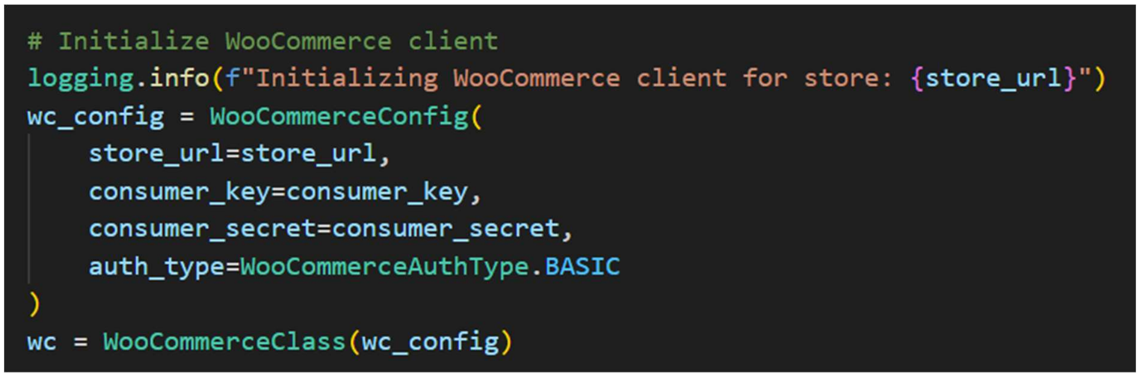
```
try:
    # Get required parameters
    logging.info("Extracting validated parameters")
    store_url = input_data.validated_data["store_url"]
    consumer_key = input_data.validated_data["consumer_key"]
    consumer_secret = input_data.validated_data["consumer_secret"]
    coupon_id = input_data.validated_data["id"]
```

b. Then get all the optional parameters



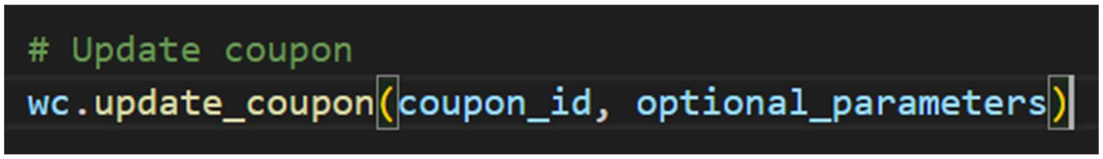
```python
# Get optional parameters
optional_parameters = {}
for param in self.get_parameter_schema():
    if param.name in input_data.validated_data:
        optional_parameters[param.name] = input_data.validated_data[param.name]
```

c. Then initialize the class



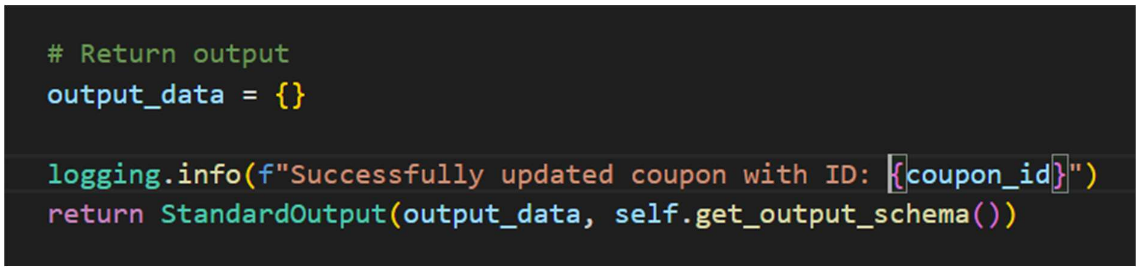
```python
# Initialize WooCommerce client
logging.info(f"Initializing WooCommerce client for store: {store_url}")
wc_config = WooCommerceConfig(
    store_url=store_url,
    consumer_key=consumer_key,
    consumer_secret=consumer_secret,
    auth_type=WooCommerceAuthType.BASIC
)
wc = WooCommerceClass(wc_config)
```

d. Then use the function from the class



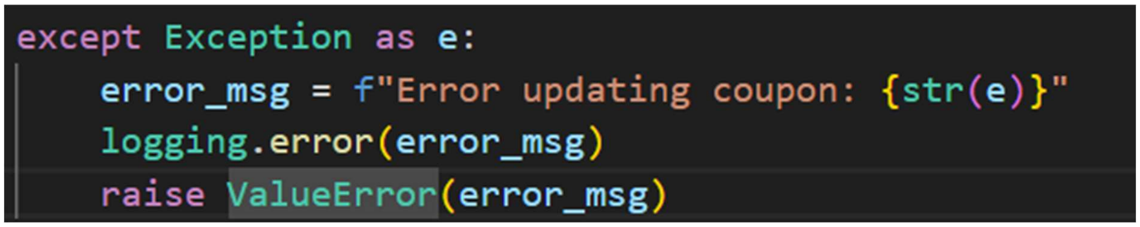
```python
# Update coupon
wc.update_coupon(coupon_id, optional_parameters)
```

e. Return nothing



```python
# Return output
output_data = {}

logging.info(f"Successfully updated coupon with ID: {coupon_id}")
return StandardOutput(output_data, self.get_output_schema())
```

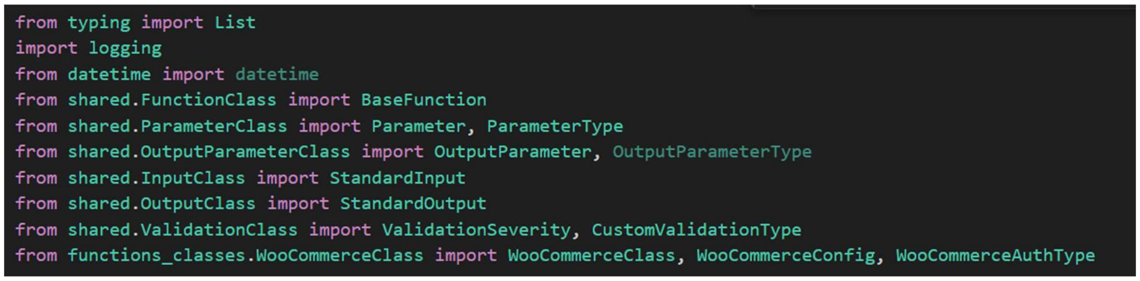
f. Make sure that all exceptions are catched



```python
except Exception as e:
    error_msg = f"Error updating coupon: {str(e)}"
    logging.error(error_msg)
    raise ValueError(error_msg)
```

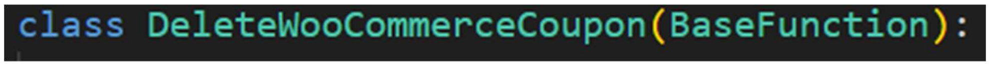
DELETE

1. First all imports



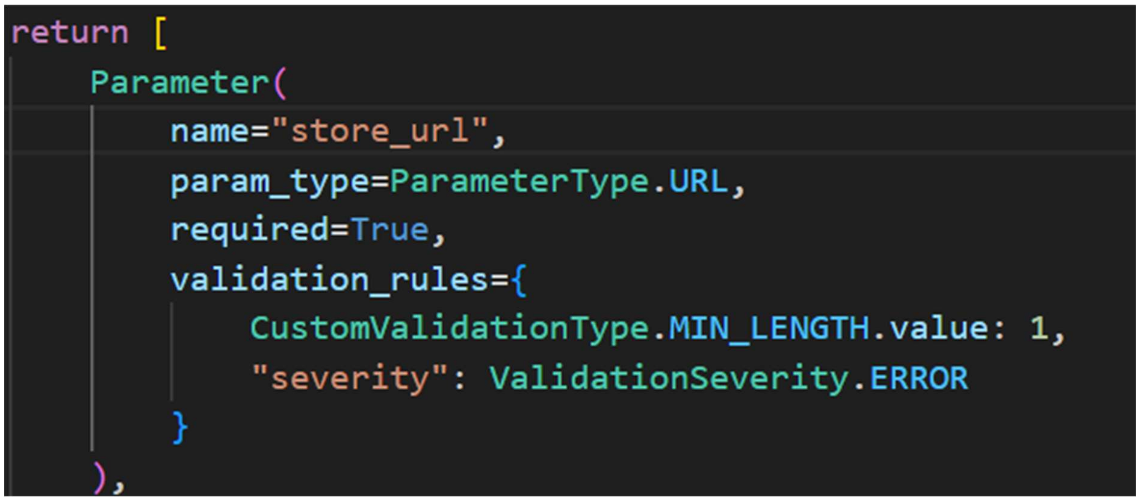
```
from typing import List
import logging
from datetime import datetime
from shared.FunctionClass import BaseFunction
from shared.ParameterClass import Parameter, ParameterType
from shared.OutputParameterClass import OutputParameter, OutputParameterType
from shared.InputClass import StandardInput
from shared.OutputClass import StandardOutput
from shared.ValidationClass import ValidationSeverity, CustomValidationType
from functions_classes.WooCommerceClass import WooCommerceClass, WooCommerceConfig, WooCommerceAuthType
```

2. Class name should be Delete{APPLICATION}{OBJECT}



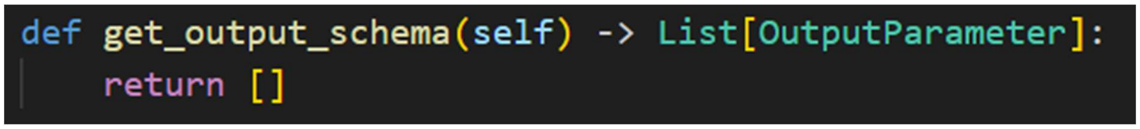
```
class DeleteWooCommerceCoupon(BaseFunction):
```

3. Generate the array of input variables (required should always be on the top)
    a. For delete besides the authentication parameters only 1 identifier should be enough (like id or slug, etc.)



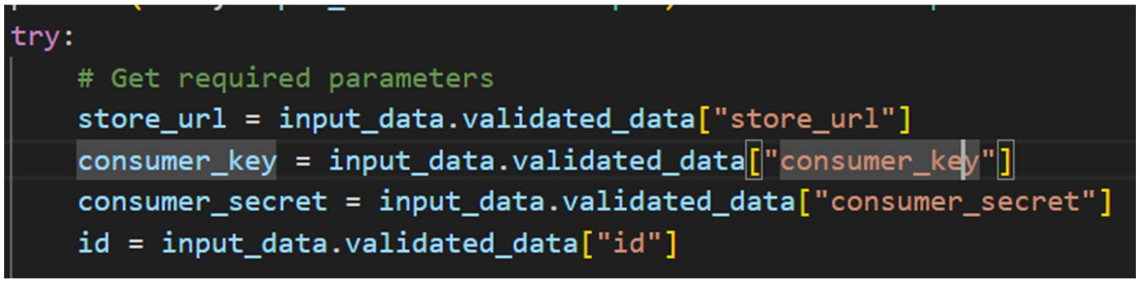
```
return [
    Parameter(
        name="store_url",
        param_type=ParameterType.URL,
        required=True,
        validation_rules={
            CustomValidationType.MIN_LENGTH.value: 1,
            "severity": ValidationSeverity.ERROR
        }
    ),
```

4. For delete nothings has to be returned



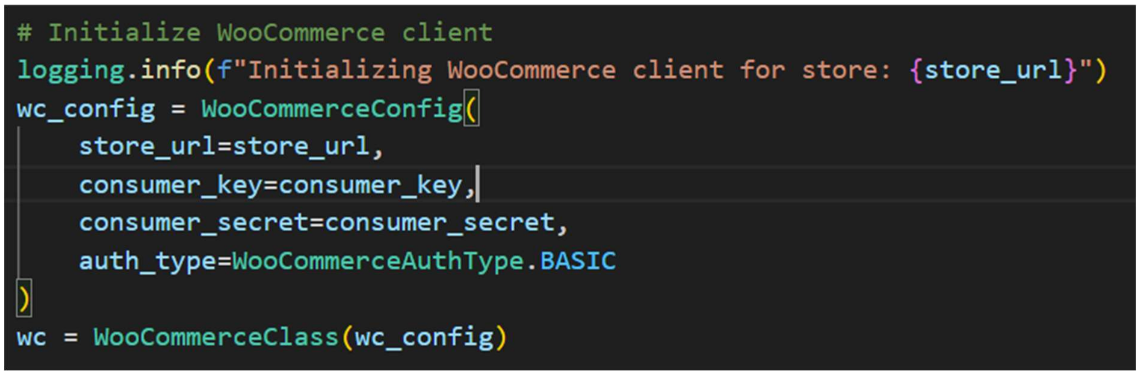
```
def get_output_schema(self) -> List[OutputParameter]:
    return []
```

5. The process function should have the implementation of the function itself
    a. Always start with getting the required parameters



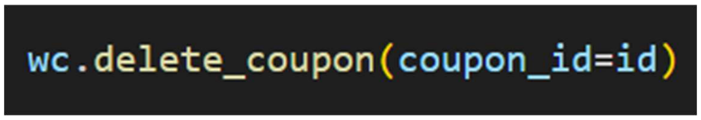
```
try:
    # Get required parameters
    store_url = input_data.validated_data["store_url"]
    consumer_key = input_data.validated_data["consumer_key"]
    consumer_secret = input_data.validated_data["consumer_secret"]
    id = input_data.validated_data["id"]
```

b. Then initialize the class



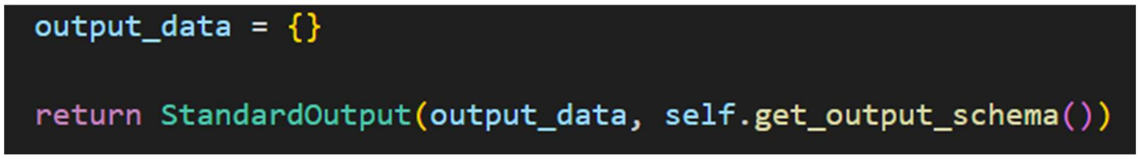
```python
# Initialize WooCommerce client
logging.info(f"Initializing WooCommerce client for store: {store_url}")
wc_config = WooCommerceConfig(
    store_url=store_url,
    consumer_key=consumer_key,
    consumer_secret=consumer_secret,
    auth_type=WooCommerceAuthType.BASIC
)
wc = WooCommerceClass(wc_config)
```

c. Then call the delete function from the class



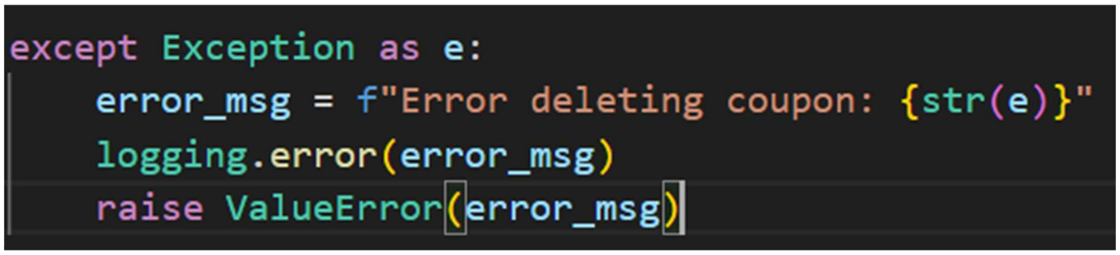
```python
wc.delete_coupon(coupon_id=id)
```

d. Return nothing



```python
output_data = {}

return StandardOutput(output_data, self.get_output_schema())
```

e. Make sure to catch all exceptions



```python
except Exception as e:
    error_msg = f"Error deleting coupon: {str(e)}"
    logging.error(error_msg)
    raise ValueError(error_msg)
```

Small remarks

- It is important that with the parameters in input all available options from the documentation are present and it does not skip any of the options which is available for that call
- The outputs of the files can become quite big so maybe it is an idea to split up the calls for the 3 different functions in each CRUD call (if you deem that necessary)