Exception Handling

Custom Exceptions - Quiz

Q1 of 8

What is the output of the below program?

```
class InvalidAccountException(Exception):
    pass
class Account:
    account_list=[1001,1002,1003,1004]
    def validate_account(self,account_id):
        status=0
        for acct_id in self.account_list:
            if(account_id==acct_id):
                status=1
                break
        if(status!=0):
            return True
        else:
            raise InvalidAccountException
try:
    account1=Account()
    account1.validate_account(1006)
    print("Valid account number")
except InvalidAccountException:
    print("Invalid account number")
```

- ○ Valid account number
- ◉ Invalid account number ✓
- ○ Error: Element not found in list

```
    def __init__(self,emp_id,emp_name):
        self.__emp_name=emp_name
        self.__emp_id=emp_id
    def validate_name(self):
        try:
            if(len(self.__emp_name) < 4):
                Employee.__trials=Employee.__trials+1
                raise NameLengthException
            if( not(self.__emp_id.startswith('E'))):
                raise EmployeeIdException
        except NameLengthException:
            Employee.__trials=Employee.__trials+1
            print(Employee.__trials)
        except EmployeeIdException:
            Employee.__trials=Employee.__trials+1
            print(Employee.__trials)
emp1=Employee('E1001','Tom')
emp1.validate_name()
emp2=Employee('1001','Tomy')
emp2.validate_name()
```

a) 2
3

b) 2
1

c) 3

d) 2

- (●) a ✓
- ( ) b
- ( ) c
- ( ) d

Q3 of 8

What will be the output of the code given below?

```
class Project:
    def __init__(self,employee_list):
        self.__employee_list=employee_list

    def validate_employee(self,employee_id):
        try:
            if employee_id not in self.__employee_list:
                raise Exception
                print("1")
        except Exception:
            print("2")

project1=Project([1001,1002,1003])
project1.validate_employee(1005)
print("3")
```

a) 2

b) 2
3

c) 2
1
3

d) 3

- ○ a
- ◉ b ✓
- ○ c
- ○ d

```
class NotEligibleException(Exception):
    pass
class Employee:
    def __init__(self,salary):
        self.__salary=salary

    def check_salary(self):
        if(self.__salary < 2000):
            raise NotEligibleException
            return False
        else:
            return True

emp1=Employee(5000)
emp2=Employee(1000)
try:
    status=emp1.check_salary()
    print(status)
    status=emp2.check_salary()
    print(status)
except NotEligibleException:
    print("Not Eligible")
```

a) True
Not Eligible

b) True
False

c) True
Not Eligible
False

- (●) a ✓
- ( ) b
- ( ) c

```
class NotEligibleException(Exception):
    pass
class Employee:
    def __init__(self,salary):
        self.__salary=salary

    def check_salary(self):
        try:
            if(self.__salary < 2000):
                raise NotEligibleException
            else:
                return True
        except NotEligibleException:
            print("1")
            raise NotEligibleException

emp1=Employee(1000)
try:
    status=emp1.check_salary()
    print("2")
except NotEligibleException:
    print("3")
```

a) 2

b) Error: An exception cannot be raised from except block of another exception

c) Error: Two exceptions (inside a method and calling block) cannot have the same name

d) 1
3

- ○ a
- ○ b
- ○ c
- ◉ d ✓

Q6 of 8

What will be the output of the code given below?

```
class InvalidEmployeeException(Exception):
    pass
class Project:
    def __init__(self,employee_list):
        self.__employee_list=employee_list

    def validate_employee(self,employee_id):
        flag=False
        for key in self.__employee_list:
            if(key==employee_id):
                flag=True
        if(flag==False):
            raise InvalidEmployeeException
            print("1")
        return True

project1=Project([1001,1002,1003])
try:
    print(project1.validate_employee(1005))
except Exception:
    print("2")
except InvalidEmployeeException:
    print("3")
```

a) 2

b) 2
3

c) 3

d) Error: Except should be the last block

(●) a ✓

( ) b

Q7 of 8

What will be the output of the code given below?

```
class CustomerBusiness:
    def get_customer(self,cust_id):
        if cust_id == "":
            raise InvalidCustomerException()
        return cust_id

class AccountUI:
    def deposit_money_ui(self):
        try:
            cust_id = CustomerBusiness().get_customer("")
        except Exception:
            print("Exception raised")
        except InvalidCustomerException:
            print("Invalid Customer Exception raised")

class InvalidCustomerException(Exception):
        pass

a=AccountUI()
a.deposit_money_ui()
```

- (●) Exception raised ✓
- ( ) Invalid Customer Exception raised
- ( ) Compile Time Error
- ( ) No Exception

```
class InvalidCustomerException(Exception):
    pass

a=AccountUI()
a.deposit_money_ui()
```

Option C

```
class CustomerBusiness:
    def get_customer(self,cust_id):
        if cust_id == None:
            raise InvalidCustomerException()
        return cust_id

class AccountUI:
    def deposit_money_ui(self):
        try:
            cust_id = CustomerBusiness().get_customer("")
        except Exception:
            print("Exception raised")
        except InvalidCustomerException:
            print("Invalid Customer Exception raised.")

class InvalidCustomerException(Exception):
    pass

a=AccountUI()
a.deposit_money_ui()
```

- ◯ Option A
- ◉ Option B ✓
- ◯ Option C