Exercise: Error Handling

Problems for exercise and homework for the Python Advanced Course @SoftUni.

1. Numbers Dictionary

You are provided with the following code:

```
numbers dictionary = {}
line = input()
while line != "Search":
    number as string = line
    number = int(input())
    numbers dictionary[number as string] = number
line = input()
while line != "Remove":
    searched = line
    print(numbers dictionary[searched])
line = input()
while line != "End":
    searched = line
    del numbers dictionary[searched]
print(numbers dictionary)
```

- On the first several lines, until you receive the command "Search", you will receive on separate lines the number as a text and the number as an integer
- When you receive "Search" on the next several lines until you receive the command "Remove", you will be given the searched number as a text, and you need to print it on the console
- When you receive "Remove" on the next several lines until you receive "End", you will be given the searched number as a text, and you need to remove it from the dictionary
- In the end, you need to **print** what is left from the **dictionary**

There is some missing code in the solution, and some errors may occur. Complete the code, so the following errors are handled:

- Passing non-integer type to the variable number
- Searching for a **non-existent** number
- Removing a non-existent number

Print appropriate messages when an error has occurred. The messages should be:















- "The variable number must be an integer"
- "Number does not exist in dictionary" for non-existing keys

Examples

Input	Output
one 1 two 2 Search one Remove two End	1 {'one': 1}
one two Search Remove End	The variable number must be an integer {}
one 1 Search one Remove two End	<pre>1 Number does not exist in dictionary {'one': 1}</pre>

2. Email Validator

You will be given some emails until you receive the command "End". Create the following custom exceptions to validate the emails:

- NameTooShortError raise it when the name in the email is less than or equal to 4 ("peter" will be the name in the email "peter@gmail.com")
- MustContainAtSymbolError raise it when there is no "@" in the email
- InvalidDomainError raise it when the domain of the email is invalid (valid domains are: .com, .bg, .net, .org)

When an error is encountered, raise it with an appropriate message:

- NameTooShortError "Name must be more than 4 characters"
- MustContainAtSymbolError "Email must contain @"
- InvalidDomainError "Domain must be one of the following: .com, .bg, .org, .net"

Hint: use the following syntax to add a message to the Exception: MyException ("Exception Message")

If the current email is valid, print "Email is valid" and read the next one











Examples

Input	Output
abc@abv.bg	<pre>Traceback (most recent call last): File ".\email_validator.py", line 20, in <module> raise NameTooShort("Name must be more than 4 characters") mainNameTooShort: Name must be more than 4 characters</module></pre>
<pre>peter@gmail.com petergmail.com</pre>	<pre>Email is valid Traceback (most recent call last): File ".\email_validator.py", line 18, in <module> raise MustContainAtSymbolError("Email must contain @") mainMustContainAtSymbolError: Email must contain @</module></pre>
peter@gmail.hotmail	<pre>Traceback (most recent call last): File ".\email_validator.py", line 22, in <module> raise InvalidDomainError("Domain must be one of the folowing: .com, .bg, .org, .net") mainInvalidDomainError: Domain must be one of the folowing: .com, .bg, .org, .net</module></pre>













