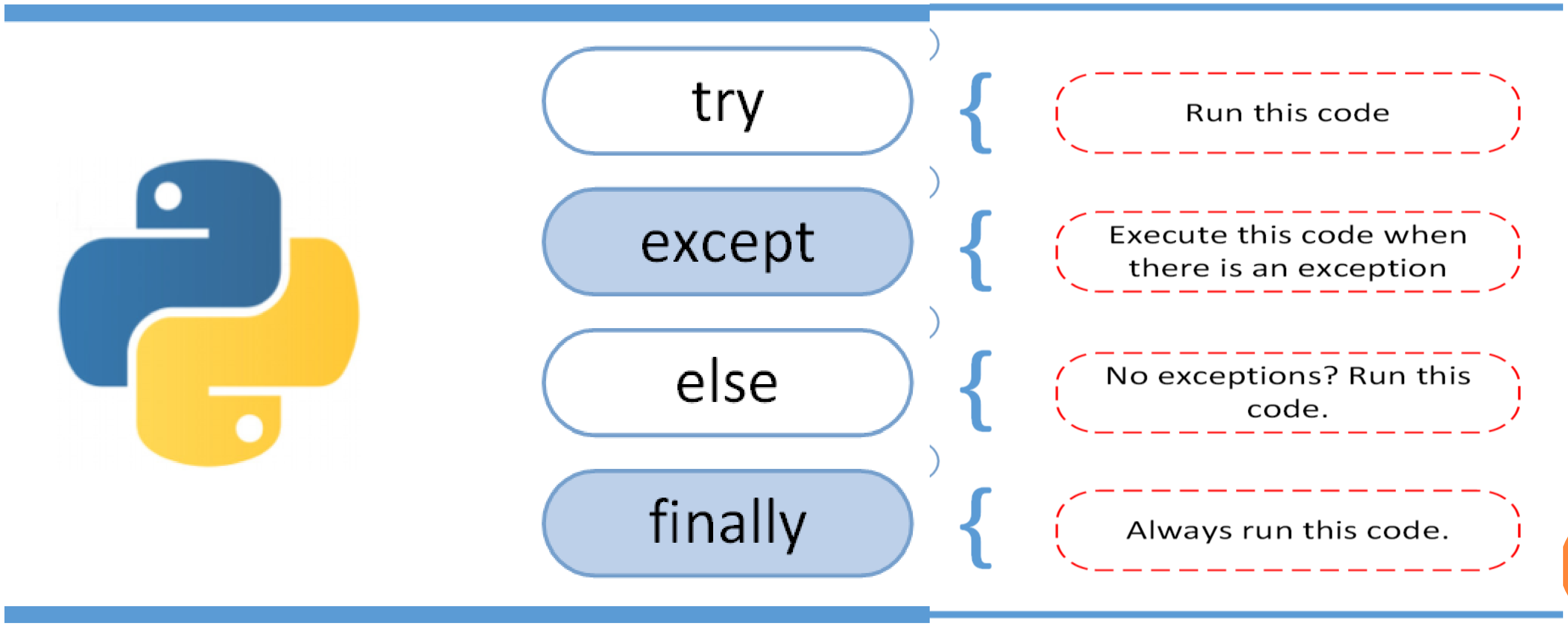
Exception classes in Python

Week 9: Exceptions

Learning objectives

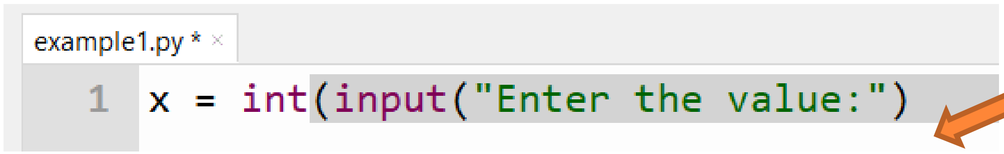
- ❑ Exception handling
- ❑ Declaring exception and distinguishing exception handling
- ❑ Implementing try-except-finally in Python

At the conclusion of this lecture, students will be able to understand the type of errors that arise in code execution and handling those errors


try
Run this code
except
Execute this code when there is an exception
else
No exceptions? Run this code.
finally
Always run this code.

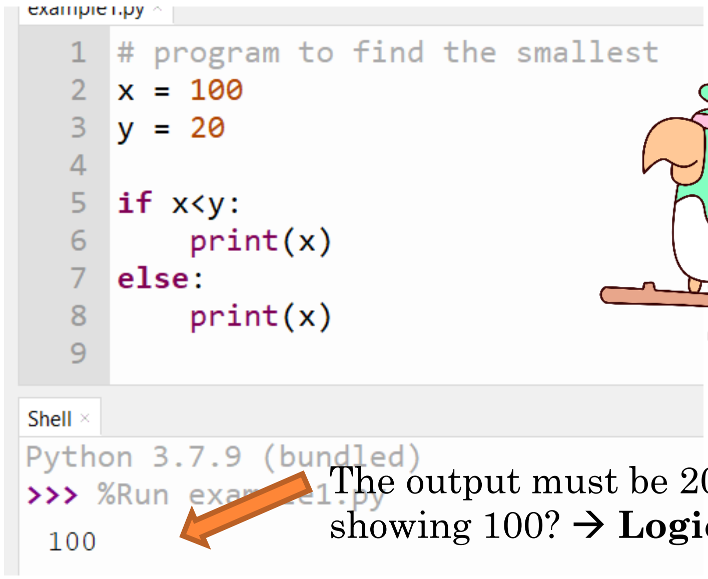
What is an error in programming?

- **These are the mistakes, problems or the faults that occur in the program.**
- Three types of errors namely: Syntax error , Logical error, and Execution/run time error



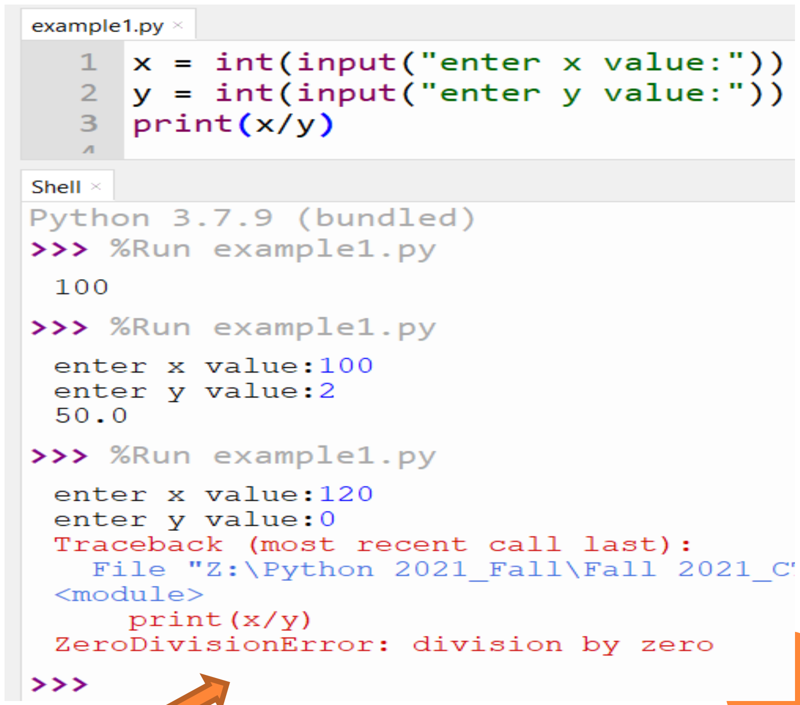
```
example1.py *
x = int(input("Enter the value:")
```

closing bracket is missing
Syntax error



```
example1.py
# program to find the smallest
x = 100
y = 20

if x<y:
    print(x)
else:
    print(x)

Shell
Python 3.7.9 (bundled)
>>> %Run example1.py
 100
```

The output must be 20 Why is it showing 100? → **Logical error**



```
example1.py
x = int(input("enter x value:"))
y = int(input("enter y value:"))
print(x/y)

Shell
Python 3.7.9 (bundled)
>>> %Run example1.py
  100
>>> %Run example1.py
  enter x value:100
  enter y value:2
  50.0
>>> %Run example1.py
  enter x value:120
  enter y value:0
  Traceback (most recent call last):
    File "Z:\Python 2021_Fall\Fall 2021_C
  <module>
      print(x/y)
  ZeroDivisionError: division by zero
>>>
```

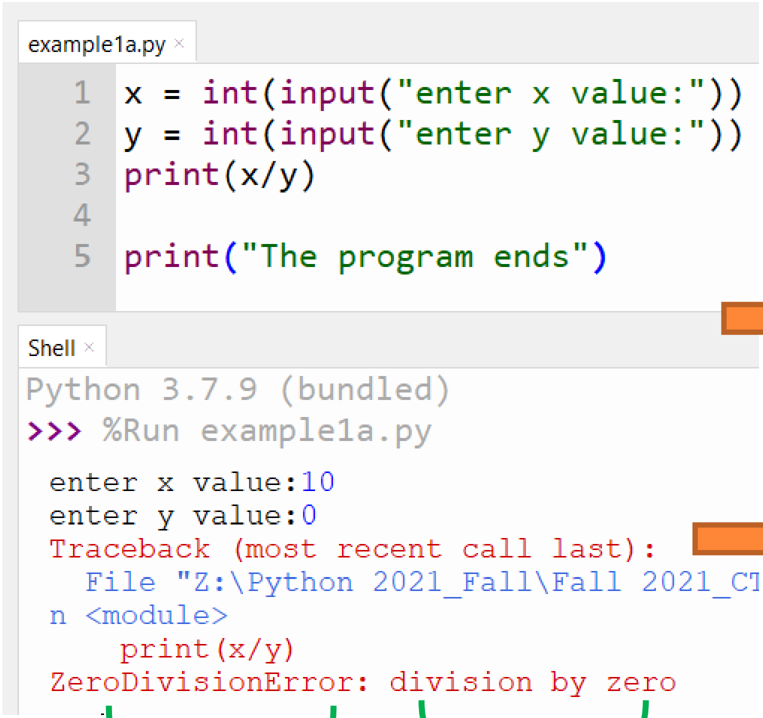
ZeroDivisionError → **Execution error**

What is Exception then?

- Errors detected **during execution** are called **exceptions**
- Example: ZeroDivisionError, FileNotFoundError, TypeError……
- Program **gets terminated abruptly** if the program is executed with execution errors.

So, how to handle possible detected errors (exception) during execution to avoid abrupt termination?

- Here, we use Python exception handling technique: **try-except-finally**



example1a.py

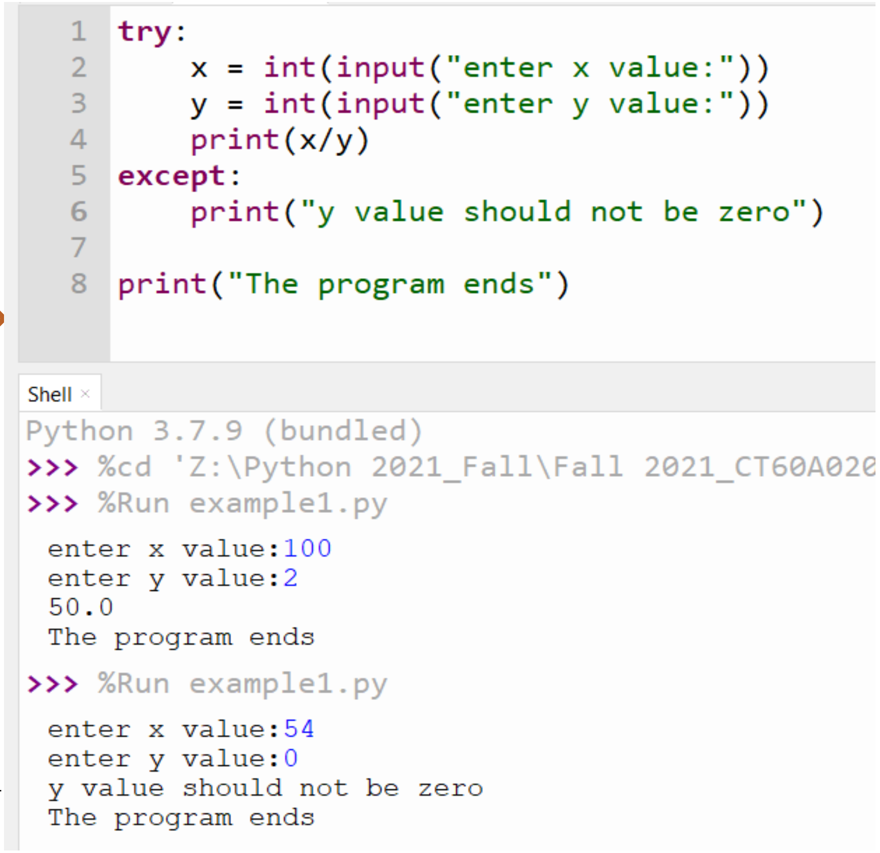
```python
x = int(input("enter x value:"))
y = int(input("enter y value:"))
print(x/y)

print("The program ends")
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run example1a.py

 enter x value:10
 enter y value:0
 Traceback (most recent call last):
   File "Z:\Python 2021_Fall\Fall 2021_CT
 n <module>
     print(x/y)
 ZeroDivisionError: division by zero
```

Name of the exception

Description of the exception

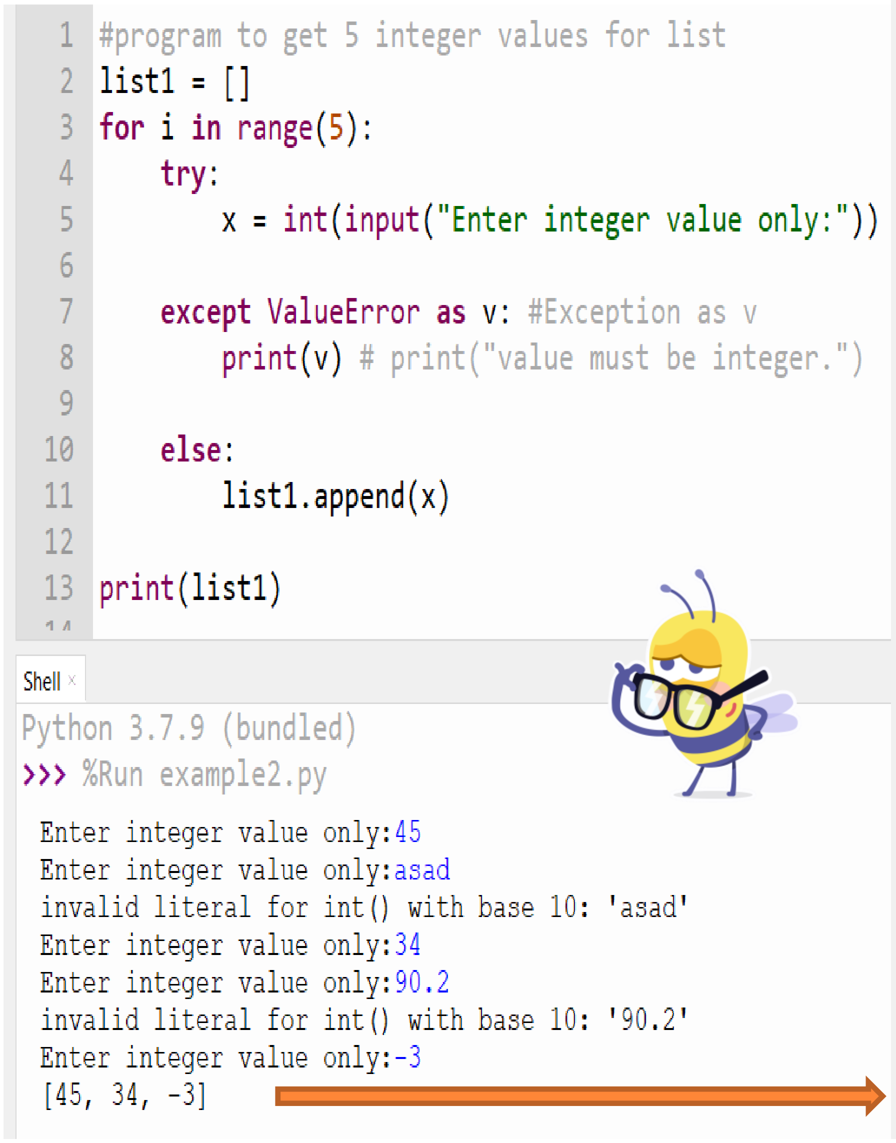


```python
try:
    x = int(input("enter x value:"))
    y = int(input("enter y value:"))
    print(x/y)
except:
    print("y value should not be zero")

print("The program ends")
```

Shell

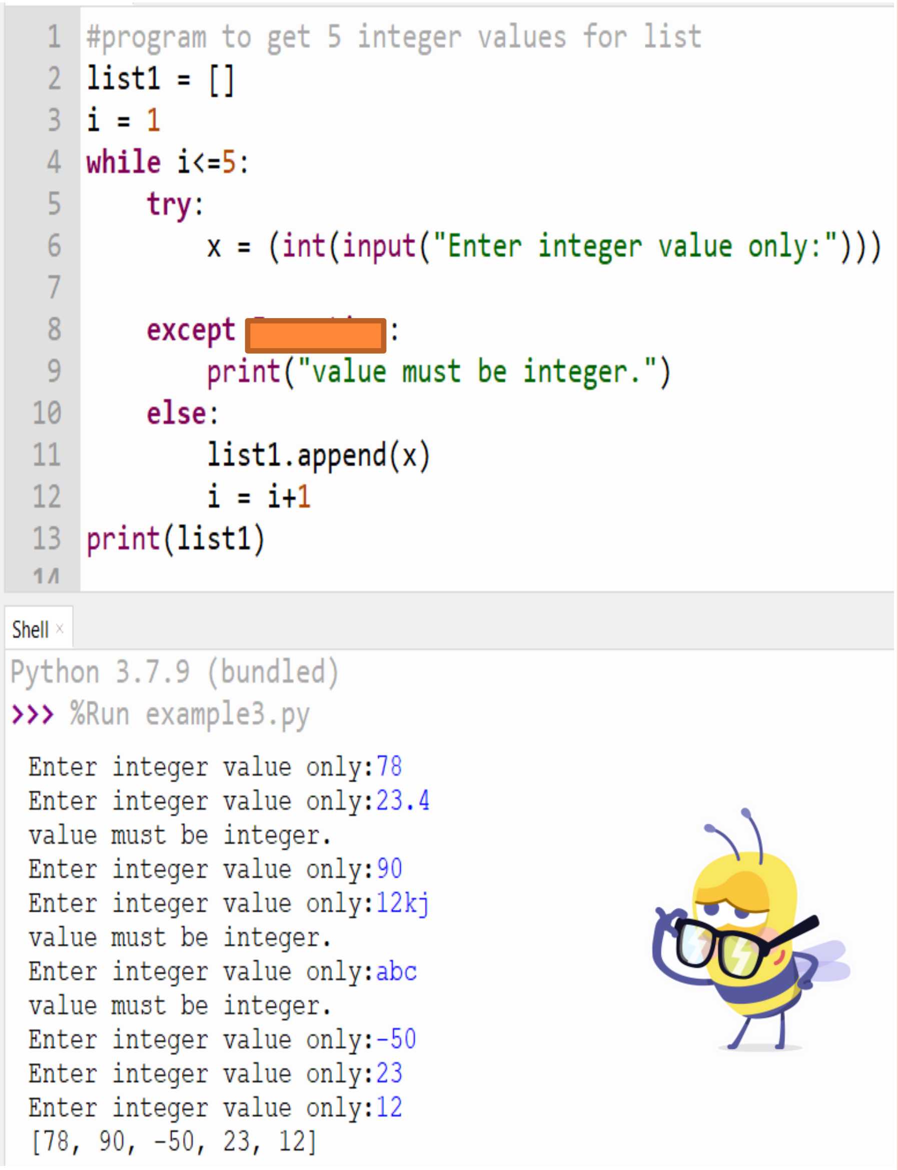
```
Python 3.7.9 (bundled)
>>> %cd 'Z:\Python 2021_Fall\Fall 2021_CT60A02
>>> %Run example1.py

 enter x value:100
 enter y value:2
 50.0
 The program ends

>>> %Run example1.py

 enter x value:54
 enter y value:0
 y value should not be zero
 The program ends
```

- **Python’s built-in Exceptions**
- Python has many built in exceptions that raised when corresponding errors occur.
- Check→ https://www.programiz.com/python-programming/exceptions



```
#program to get 5 integer values for list
list1 = []
for i in range(5):
    try:
        x = int(input("Enter integer value only:"))

    except ValueError as v: #Exception as v
        print(v) # print("value must be integer.")

    else:
        list1.append(x)

print(list1)
```

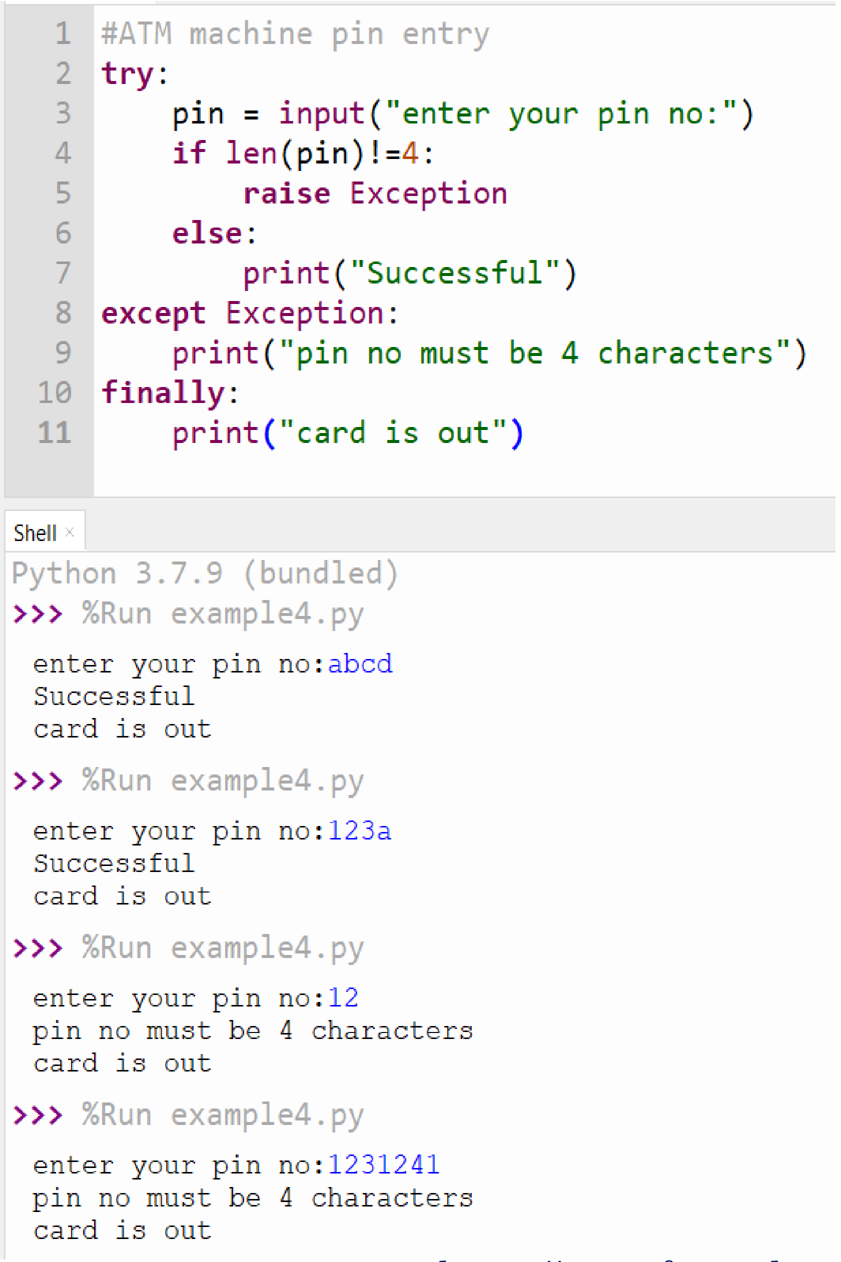
```
Shell

Python 3.7.9 (bundled)
>>> %Run example2.py

 Enter integer value only:45
 Enter integer value only:asad
 invalid literal for int() with base 10: 'asad'
 Enter integer value only:34
 Enter integer value only:90.2
 invalid literal for int() with base 10: '90.2'
 Enter integer value only:-3
 [45, 34, -3]
```



```
#program to get 5 integer values for list
list1 = []
i = 1
while i<=5:
    try:
        x = (int(input("Enter integer value only:")))

    except [redacted]:
        print("value must be integer.")
    else:
        list1.append(x)
        i = i+1
print(list1)
```

```
Shell

Python 3.7.9 (bundled)
>>> %Run example3.py

 Enter integer value only:78
 Enter integer value only:23.4
 value must be integer.
 Enter integer value only:90
 Enter integer value only:12kj
 value must be integer.
 Enter integer value only:abc
 value must be integer.
 Enter integer value only:-50
 Enter integer value only:23
 Enter integer value only:12
 [78, 90, -50, 23, 12]
```

try-except-else-finally

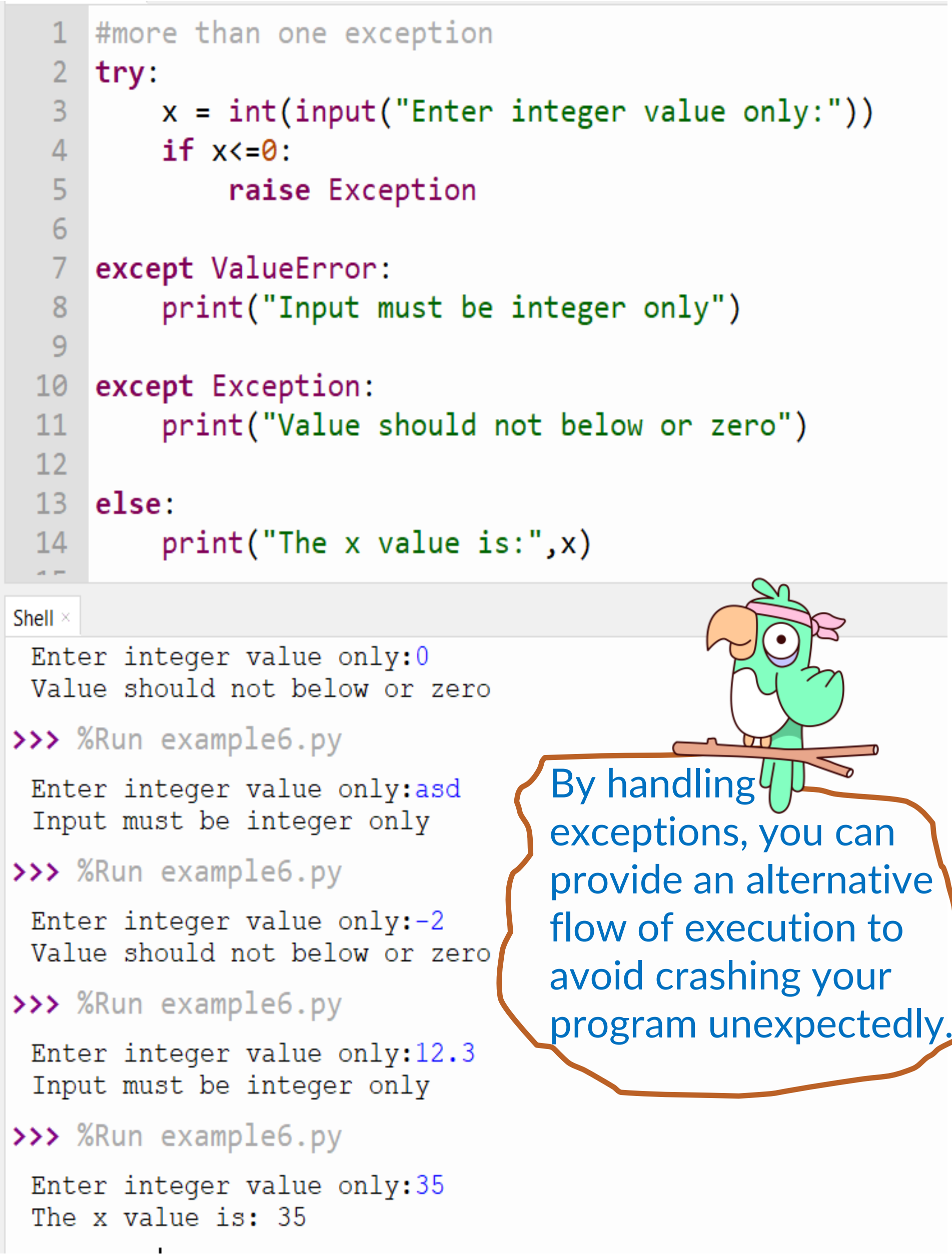


```
#ATM machine pin entry
try:
    pin = input("enter your pin no:")
    if len(pin)!=4:
        raise Exception
    else:
        print("Successful")
except Exception:
    print("pin no must be 4 characters")
finally:
    print("card is out")
```

Shell

```
Python 3.7.9 (bundled)
>>> %Run example4.py

 enter your pin no:abcd
 Successful
 card is out

>>> %Run example4.py

 enter your pin no:123a
 Successful
 card is out

>>> %Run example4.py

 enter your pin no:12
 pin no must be 4 characters
 card is out

>>> %Run example4.py

 enter your pin no:1231241
 pin no must be 4 characters
 card is out
```

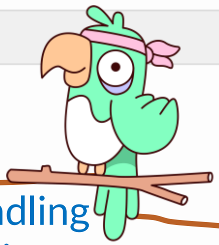
Catching more than one Exception



```
#more than one exception
try:
    x = int(input("Enter integer value only:"))
    if x<=0:
        raise Exception

except ValueError:
    print("Input must be integer only")

except Exception:
    print("Value should not below or zero")

else:
    print("The x value is:",x)
```

Shell

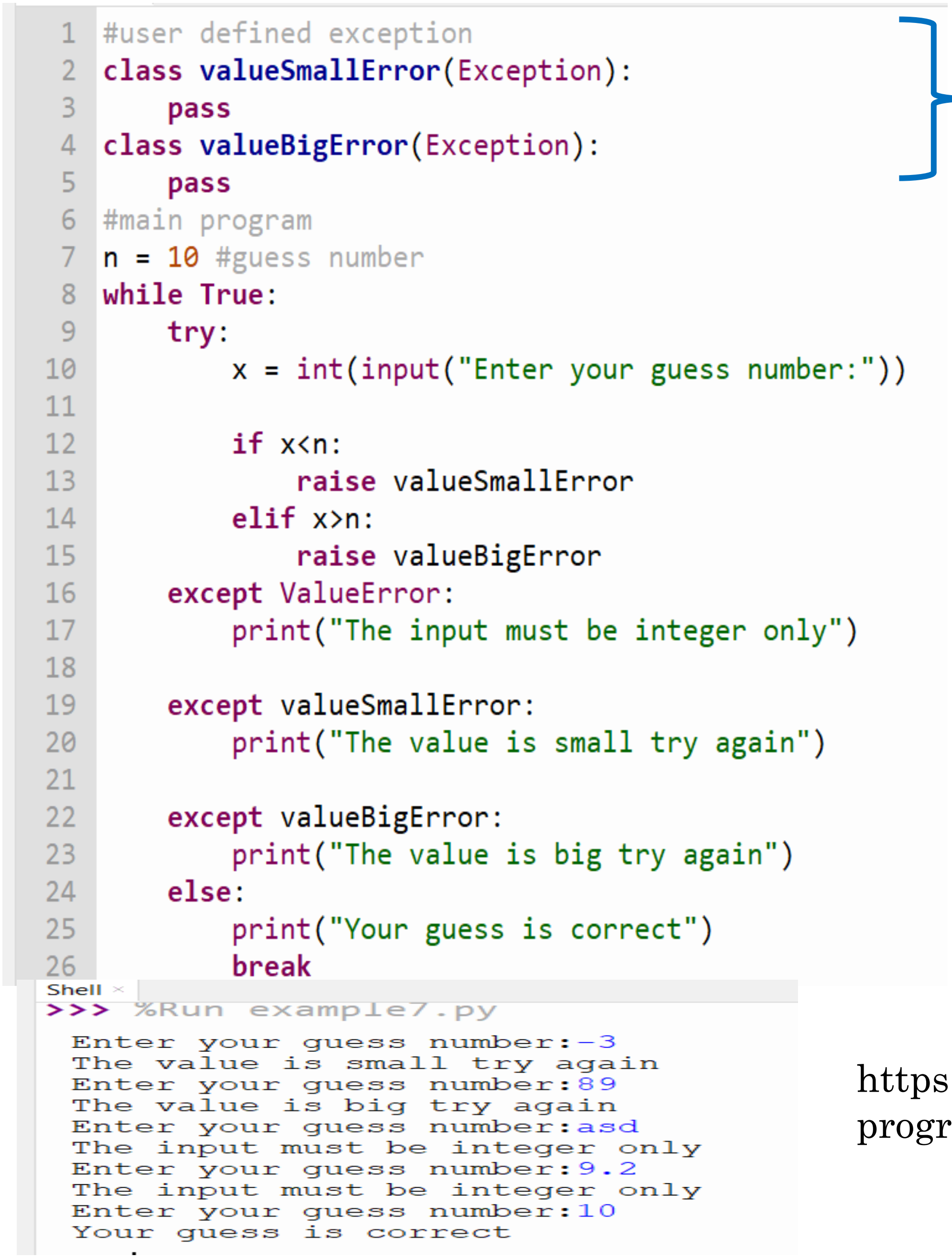
```
 Enter integer value only:0
 Value should not below or zero

>>> %Run example6.py

 Enter integer value only:asd
 Input must be integer only

>>> %Run example6.py

 Enter integer value only:-2
 Value should not below or zero

>>> %Run example6.py

 Enter integer value only:12.3
 Input must be integer only

>>> %Run example6.py

 Enter integer value only:35
 The x value is: 35
```



By handling exceptions, you can provide an alternative flow of execution to avoid crashing your program unexpectedly.

https://www.freecodecamp.org/news/exception-handling-python/

User defined (custom) exception



```
#user defined exception
class valueSmallError(Exception):
    pass
class valueBigError(Exception):
    pass
#main program
n = 10 #guess number
while True:
    try:
        x = int(input("Enter your guess number:"))

        if x<n:
            raise valueSmallError
        elif x>n:
            raise valueBigError
    except ValueError:
        print("The input must be integer only")

    except valueSmallError:
        print("The value is small try again")

    except valueBigError:
        print("The value is big try again")
    else:
        print("Your guess is correct")
        break
```

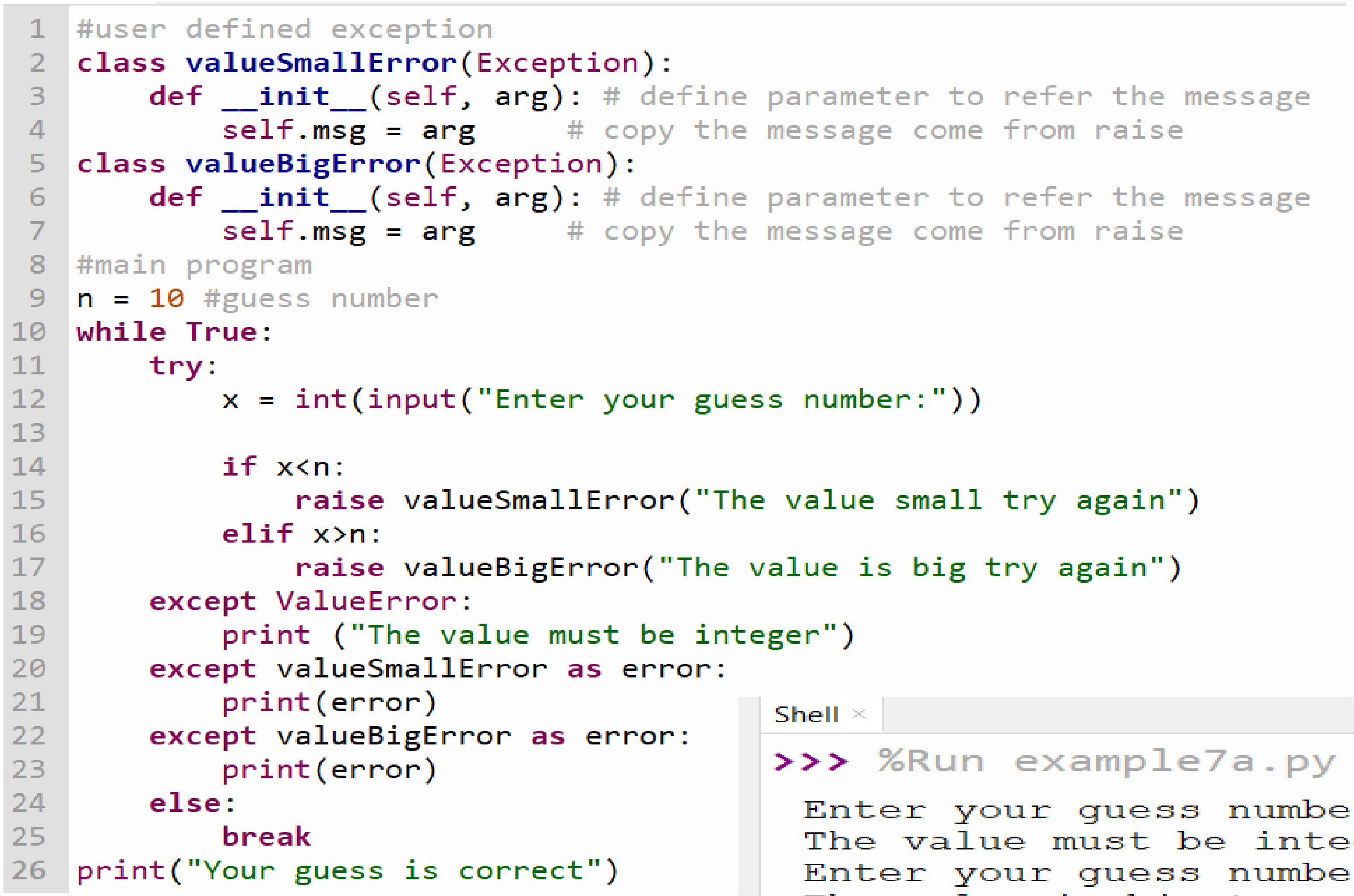
Custom/user defined exception

```
>>> %Run example7.py

  Enter your guess number:-3
  The value is small try again
  Enter your guess number:89
  The value is big try again
  Enter your guess number:asd
  The input must be integer only
  Enter your guess number:9.2
  The input must be integer only
  Enter your guess number:10
  Your guess is correct
```

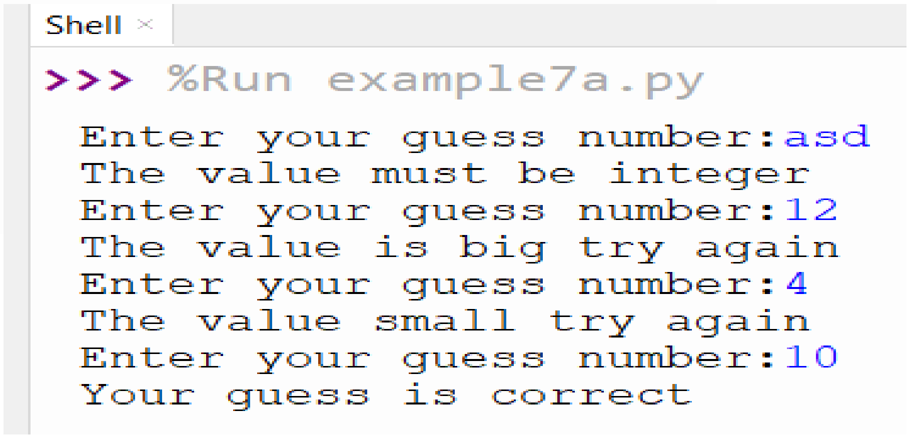
pass used when a statement is required syntactically but you do not want any command or code to execute. It is like null operation, as nothing will happen is it is executed. Pass statement can also be used for writing empty loops.

https://www.programiz.com/python-programming/user-defined-exception

User defined (custom) exception



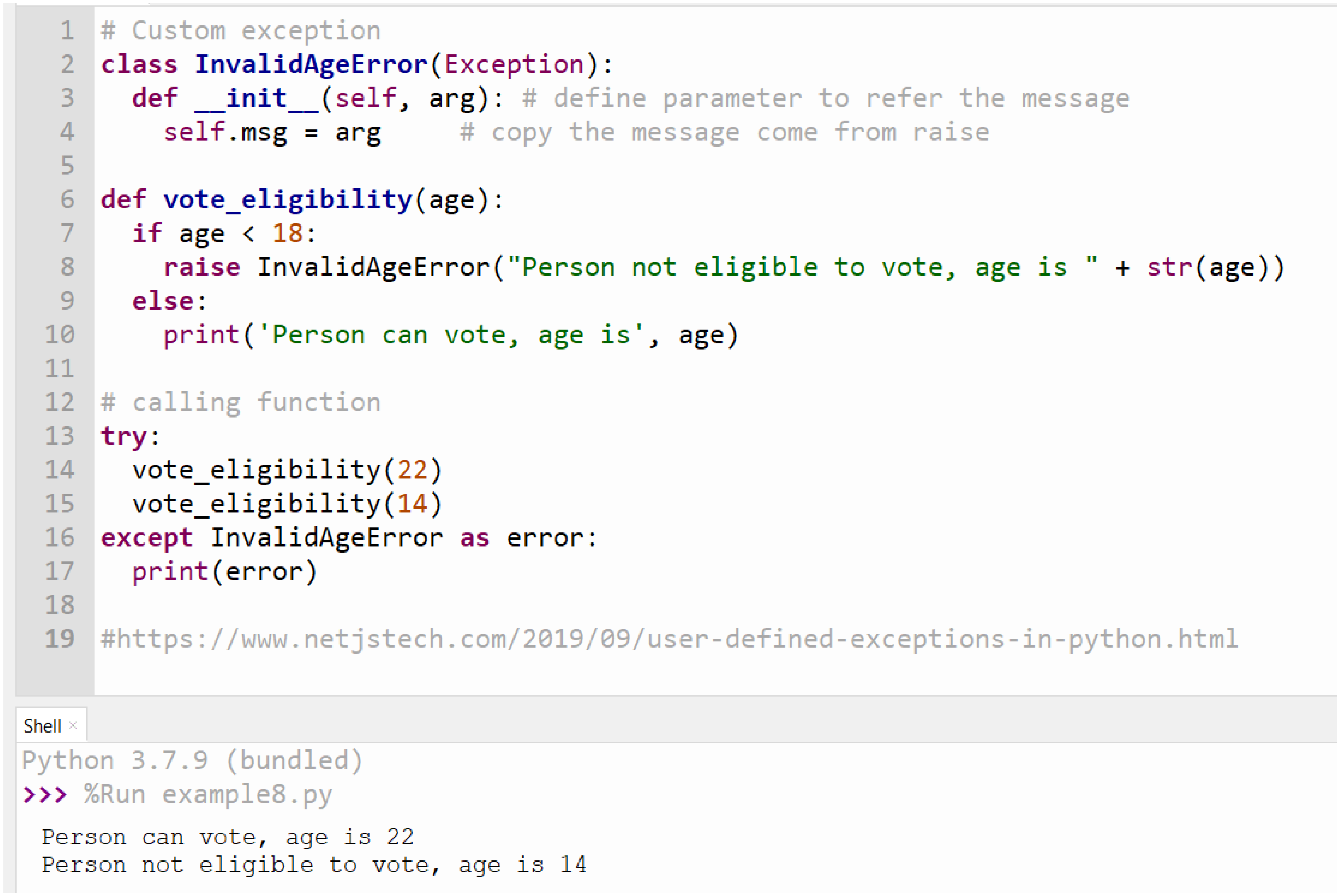
```python
#user defined exception
class valueSmallError(Exception):
    def __init__(self, arg): # define parameter to refer the message
        self.msg = arg      # copy the message come from raise
class valueBigError(Exception):
    def __init__(self, arg): # define parameter to refer the message
        self.msg = arg      # copy the message come from raise
#main program
n = 10 #guess number
while True:
    try:
        x = int(input("Enter your guess number:"))

        if x<n:
            raise valueSmallError("The value small try again")
        elif x>n:
            raise valueBigError("The value is big try again")
    except ValueError:
        print ("The value must be integer")
    except valueSmallError as error:
        print(error)
    except valueBigError as error:
        print(error)
    else:
        break
print("Your guess is correct")
```



Shell

```
>>> %Run example7a.py

 Enter your guess number:asd
 The value must be integer
 Enter your guess number:12
 The value is big try again
 Enter your guess number:4
 The value small try again
 Enter your guess number:10
 Your guess is correct
```

User defined (custom) exception



```python
# Custom exception
class InvalidAgeError(Exception):
  def __init__(self, arg): # define parameter to refer the message
    self.msg = arg      # copy the message come from raise

def vote_eligibility(age):
  if age < 18:
    raise InvalidAgeError("Person not eligible to vote, age is " + str(age))
  else:
    print('Person can vote, age is', age)

# calling function
try:
  vote_eligibility(22)
  vote_eligibility(14)
except InvalidAgeError as error:
  print(error)

#https://www.netjstech.com/2019/09/user-defined-exceptions-in-python.html
```

```
Python 3.7.9 (bundled)
>>> %Run example8.py

 Person can vote, age is 22
 Person not eligible to vote, age is 14
```