Exceptions

- An exception is an error which occurs while a program is running.

- try-except statement:
  - monitor code that could produce an error
  - provide error-specific recovery code

```
try:
    suite of code

except SomeException:
    suite to handle specific exception
```
Exceptions

- An exception is an error which occurs while a program is running.

- **try-except statement:**
  - monitor code that could produce an error
  - provide error-specific recovery code

```
try:
    suite of code

except SomeException:
    suite to handle specific exception
```

- **Try block** (code to monitor)
- **Except block** (exception handler)
Can have multiple except suites and/or one unnamed except suite

```
try:
    suite of code

except SomeException:
    suite to handle SomeException

except AnotherException:
    suite to handle AnotherException

...

except:
    suite to handle any other exception
```
Can have multiple except suites and/or one unnamed except suite

```
try:
    suite of code

except SomeException:
    suite to handle SomeException

except AnotherException:
    suite to handle AnotherException

. . .

except:
    suite to handle any other exception
```

Try block (code to monitor)

Except blocks (exception handlers)
Flow of control

Execute the try suite line-by-line
  o if no errors occurs, skip all except blocks
  o if an error occurs and one of the try blocks handles that kind of error
    • stop execution of the try block and execute the except block
    • skip past all other except blocks, if any
  o if an error occurs and none of the try blocks handles that kind of error, *propagate* the error:
    • terminate the try-except statement and re-raises the exception
    • if in a function call, raise the exception in the caller
    • if at top level (global frame), terminate the program
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom
except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0
except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0
print( "10 divided by", denom, "yields", result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: " )
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator:")
    denom = int( value )
    print( "Denominator:", denom )
    result = 10/denom
except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0
except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0
print( "10 divided by", denom, "yields", result )
print( "Program halted." )
Assuming user inputs “5.0”:

```python
try:
    value = input( "Enter a denominator: ")
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom , "yields" , result )
print( "Program halted." )
```
try:
    value = input( "Enter a denominator: ")
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
try:
    value = input( "Enter a denominator: ")
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by zero." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom , "yields" , result )
print( "Program halted." )
try:
value = input( "Enter a denominator: ")
denom = int( value )
print( "Denominator:" , denom )
result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by 0." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by", denom, "yields", result )
print( "Program halted." )
Assuming user inputs “0”:

```
try:
    value = input( "Enter a denominator: ")
    denom = int( value )
    print( "Denominator:" , denom )
    result = 10/denom

except ValueError:
    print( "Unable to convert input to type int." )
    print( "Denominator set to 2." )
    denom, result = 2, 5.0

except ZeroDivisionError:
    print( "Invalid: attempted to divide by 0." )
    print( "Denominator set to 1." )
    denom, result = 1, 10.0

print( "10 divided by" , denom, "yields" , result )
print( "Program halted." )
```
Idiom – repeatedly prompt if file not found

```python
while True:
    name = input( "Enter a file name: " )

    try:
        file = open( name, "r" )
        break

    except FileNotFoundError:
        print( "Invalid file name, try again" )

# code to read from file and process data
.
.
.
file.close()
```
What exceptions are there?

- To find the exception for a specific case, simply create the error in the shell and see what appears in the error message (or consult the Python documentation).

- Check for specific exceptions
  - An unnamed except block will catch anything.
  - Not a good programming practice: name the exceptions which you think might occur.