COL1000
Introduction to Programming

Priyanka Golia

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides

```
print(10 / 0)
int("hello")
nums = [1, 2, 3]
print(nums[10])
```

```
print(10 / 0)          # ZeroDivisionError
int("hello")           # ValueError
nums = [1, 2, 3]
print(nums[10])        # IndexError
```

```
print(10 / 0)          # ZeroDivisionError
int("hello")           # ValueError
nums = [1, 2, 3]
print(nums[10])        # IndexError
```

Why does the program *stop* rather than continuing?

```
print(10 / 0)          # ZeroDivisionError
int("hello")           # ValueError
nums = [1, 2, 3]
print(nums[10])        # IndexError
```

Why does the program *stop* rather than continuing?

It has raised an **exception**—a signal that "something abnormal happened." If no one *handles* it, Python stops and shows a *traceback* so you can see where the problem originated.

Exceptions!

```
print(10 / 0)          # ZeroDivisionError
int("hello")           # ValueError
nums = [1, 2, 3]
print(nums[10])        # IndexError
```

Why does the program *stop* rather than continuing?

It has raised an **exception**—a signal that "something abnormal happened." If no one *handles* it, Python stops and shows a *traceback* so you can see where the problem originated.

“try ... except”

- A banking program dividing by zero (balance/number_of_months).
- A file-processing tool reading 1 —1000 files—one missing file shouldn’t kill the whole batch.
- A marks calculator reading text instead of numbers from user input.

We want a way to **gracefully handle** problems rather than letting the program crash.

That’s the purpose of the try — except mechanism.

“try ... except”

```
try:
    # code that *might* fail
    n = int(input("Enter a number: "))
    print(10 / n)
except ValueError:
    print("That was not an integer.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

If input an integer except 0 and string.

If input is “hello”, “abc” ....

If input is 0....

“try ... except”

```
try:
    # code that *might* fail
    n = int(input("Enter a number: "))
    print(10 / n)
except ValueError:
    print("That was not an integer.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

If input an integer except 0 and string.

If input is “hello”, “abc” ....

If input is 0....

Python runs everything inside try ..

If an error occurs, it *jumps out* to the first matching except.

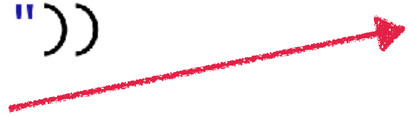
If none matches, the program ends with an error message.

“try ... except”

```
try:
    n = int(input("Enter a number: "))
    while n != "end":
        print(10 / n)
        n = int(input("Enter a number: "))
except ValueError:
    print("That was not an integer.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

“try ... except”

```
try:
    n = int(input("Enter a number: "))
    while n != "end":
        print(10 / n)
        n = int(input("Enter a number: "))
except ValueError:
    print("That was not an integer.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```



Get out of loop as soon as exception is raised.

“try ... except”

```
try:
    n = int(input("Enter a number: "))
    while n != "end":
        print(10 / n)
        n = int(input("Enter a number: "))
except ValueError:
    print("That was not an integer.")
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

Get out of loop as soon as exception is raised.

```
Enter a number: 5
2.0
Enter a number: 10
1.0
Enter a number: 0
Cannot divide by zero.
```

“try ... except”

```
n = input("enter a number")
while n!= "end":
    try:
        n = int(n)
        print(10/n)
    except ValueError:
        print("that is not an interger")
    except ZeroDivisionError:
        print("cann't divide by 0")
    n = input("enter another number or \' end \' to stop")
if n == "end":
    print("program end")
```

“try ... except”

```
1  n = input("enter a number")
2  while n!= "end":
3      try:
4          n = int(n)
5          print(10/n)
6      except ValueError:
7          print("that is not an interger")
8      except ZeroDivisionError:
9          print("cann't divide by 0")
10     n = input("enter another number or \' end \' to stop")
11 if n == "end":
12     print("program end")
```

Program raises exception and continues.

“try ... except”

```
1  n = input("enter a number")
2  while n!= "end":
3      try:
4          n = int(n)
5          print(10/n)
6      except ValueError:
7          print("that is not an interger")
8      except ZeroDivisionError:
9          print("cann't divide by 0")
10     n = input("enter another number or \' end \' to stop")
11 if n == "end":
12     print("program end")
```

Program raises exception and continues.

```
enter a number3
3.3333333333333335
enter another number or ' end ' to stop5
2.0
enter another number or ' end ' to stopabc
that is not an interger
enter another number or ' end ' to stop0
cann't divide by 0
enter another number or ' end ' to stop4
2.5
enter another number or ' end ' to stopend
program end
```

“try ... except”

```
1  n = input("enter a number")
2  while n!= "end":
3      try:
4          n = int(n)
5          print(10/n)
6      except ValueError:
7          print("that is not an interger")
8      except ZeroDivisionError:
9          print("cann't divide by 0")
10     n = input("enter another number or \' end \' to stop")
11 if n == "end":
12     print("program end")
```

Program raises exception and continues.

except (ZeroDivisionError, ValueError):
    print("Bad input of some kind.")

```
enter a number3
3.3333333333333335
enter another number or ' end ' to stop5
2.0
enter another number or ' end ' to stopabc
that is not an interger
enter another number or ' end ' to stop0
cann't divide by 0
enter another number or ' end ' to stop4
2.5
enter another number or ' end ' to stopend
program end
```

Raising exceptions yourself

```
1 def grade(score):
2     if not (0 <= score <= 100):
3         raise ValueError(f"Invalid score: {score}")
4     if score >= 90: return "A"
5     if score >= 80: return "B"
6     if score >= 70: return "C"
7     return "D"
8
9 print(grade(95))
10 print(grade(35))
11 print(grade(150))
```

Raising exceptions yourself

```
1  def grade(score):
2      if not (0 <= score <= 100):
3          raise ValueError(f"Invalid score: {score}")
4      if score >= 90: return "A"
5      if score >= 80: return "B"
6      if score >= 70: return "C"
7      return "D"
8
9  print(grade(95))
10 print(grade(35))
11 print(grade(150))
```

you can throw an exception when your function detects an invalid situation.

Raising exceptions yourself

```
1  def grade(score):
2      if not (0 <= score <= 100):
3          raise ValueError(f"Invalid score: {score}")
4      if score >= 90: return "A"
5      if score >= 80: return "B"
6      if score >= 70: return "C"
7      return "D"
8
9  print(grade(95))
10 print(grade(35))
11 print(grade(150))
```

Notice it is “function” type object. Callable object. Calling it with msg.

you can throw an exception when your function detects an invalid situation.

Raising exceptions yourself

```python
def grade(score):
    if not (0 <= score <= 100):
        raise ValueError(f"Invalid score: {score}")
    if score >= 90: return "A"
    if score >= 80: return "B"
    if score >= 70: return "C"
    return "D"

print(grade(95))
print(grade(35))
print(grade(150))
```

Notice it is “function” type object. Callable object. Calling it with msg.

you can throw an exception when your function detects an invalid situation.

```
A
D
Traceback (most recent call last):
  File "run.py", line 1, in <module>
    import lec_main
  File "/home/p11208/lec_main.py", line 3, in <module>
    import lec25
  File "/home/p11208/lec25.py", line 11, in <module>
    print(grade(150))
  File "/home/p11208/lec25.py", line 3, in grade
    raise ValueError(f"Invalid score: {score}")
ValueError: Invalid score: 150
```

“try ... except”

Some built-in Exceptions:

- TypeError
- ValueError
- NameError
- IndexError
- KeyError
- ZeroDivisionError
- KeyboardInterrupt
- AssertionError
- ModuleNotFoundError
- ImportError

“try ... except”

Some built-in Exceptions:

- TypeError
- ValueError
- NameError
- IndexError
- KeyError
- ZeroDivisionError
- KeyboardInterrupt
- AssertionError
- ModuleNotFoundError
- ImportError

Or you can used just “Exception”

```
1 n = input("enter a number")
2 while n!="end":
3     try:
4         n = int(n)
5         print(10/n)
6     except Exception as e:
7         print("exception",e)
8     n = input("enter another number or \'end\'")
```

“try ... except”

Some built-in Exceptions:

- TypeError
- ValueError
- NameError
- IndexError
- KeyError
- ZeroDivisionError
- KeyboardInterrupt
- AssertionError
- ModuleNotFoundError
- ImportError

Or you can used just “Exception”

```
1 n = input("enter a number")
2 while n!="end":
3     try:
4         n = int(n)
5         print(10/n)
6     except Exception as e:
7         print("exception",e)
8     n = input("enter another number or \'end\'")
```

```
enter a number4
2.5
enter another number or 'end'3
3.3333333333333335
enter another number or 'end'1
10.0
enter another number or 'end'abc
exception invalid literal for int() with base 10: 'abc
enter another number or 'end'0
exception division by zero
enter another number or 'end'
```

“try ... except”

```
try:
    n = input("enter a number")
    n = int(n)
    e = 10/n
except ValueError as v:
    print("that is not an integer", v)
except ZeroDivisionError as z:
    print("can not divide by zero", z)
else:
    print(f"output is {10/n}")
finally:
    print("program ends")
```

“try ... except”

```
try:
    n = input("enter a number")
    n = int(n)
    e = 10/n
except ValueError as v:
    print("that is not an integer", v)
except ZeroDivisionError as z:
    print("can not divide by zero", z)
else:
    print(f"output is {10/n}")
finally:
    print("program ends")
```

Clause	When it runs	Typical purpose
try	Normal code	Attempt the risky work
except	If an exception occurs	Handle / recover
else	If no exception occurred	Code that should run only on success
finally	Always	Cleanup (closing files, releasing resources)

“try ... except”

```
1  try:
2      n = input("enter a number")
3      n = int(n)
4      e = 10/n
5  except ValueError as v:
6      print("that is not an integer", v)
7  except ZeroDivisionError as z:
8      print("can not divide by zero", z)
9  else:
10     print(f"output is {10/n}")
11 finally:
12     print("program ends")
```

```
enter a number4
output is 2.5
program ends
```

```
enter a number0
can not divide by zero division by zero
program ends
```

```
enter a numberabc
that is not an integer invalid literal for int() with
program ends
```

Try .. Except

```python
def risky_division(a, b):
    try:
        print("Inner try started.")
        return a / b
    except ZeroDivisionError:
        print("Inner except: divide by zero.")
        return None
    finally:
        print("Inner finally executed.")
try:
    print("Outer try started.")
    result = risky_division(10, 0)
    print("Result:", result)
except Exception as e:
    print("Outer except got:", e)
finally:
    print("Outer finally executed.")
```

Try .. Except

```python
def risky_division(a, b):
    try:
        print("Inner try started.")
        return a / b
    except ZeroDivisionError:
        print("Inner except: divide by zero.")
        return None
    finally:
        print("Inner finally executed.")
try:
    print("Outer try started.")
    result = risky_division(10, 0)
    print("Result:", result)
except Exception as e:
    print("Outer except got:", e)
finally:
    print("Outer finally executed.")
```

```
Outer try started.
Inner try started.
Inner except: divide by zero.
Inner finally executed.
Result: None
Outer finally executed.
```

Try .. Except

```python
def risky_division(a, b):
    try:
        print("Inner try started.")
        return a / b
    except ZeroDivisionError:
        print("Inner except: divide by zero.")
        return None
    finally:
        print("Inner finally executed.")
try:
    print("Outer try started.")
    result = risky_division(10, 0)
    print("Result:", result)
except Exception as e:
    print("Outer except got:", e)
finally:
    print("Outer finally executed.")
```

Step	Context	Code executed	Output
1	outer	print("Outer try started.")	Outer try started.
2	outer → inner	print("Inner try started.")	Inner try started.
3	inner	a / b → raises ZeroDivisionError	—
4	inner except	print("Inner except...")	Inner except: divide by zero.
5	inner finally	print("Inner finally executed.")	Inner finally executed.
6	function returns	returns `None`	—
7	outer	print("Result:", result)	Result: None
8	outer finally	print("Outer finally executed.")	Outer finally executed.

```
Outer try started.
Inner try started.
Inner except: divide by zero.
Inner finally executed.
Result: None
Outer finally executed.
```

Try .. Except

```python
def f3(): return 10 / 0
def f2(): f3()
def f1():
    try:
        f2()
    except ZeroDivisionError as e:
        print("Caught in f1:", e)

f1()
```

```
Caught in f1: division by zero
```

f3: no return value is produced; it exits via exception.

f2: no return value is produced; it also exits via the same exception.

f1: the exception is caught; the except block runs and prints the message. After the except block, f1 completes normally (no explicit return), so f1 returns None