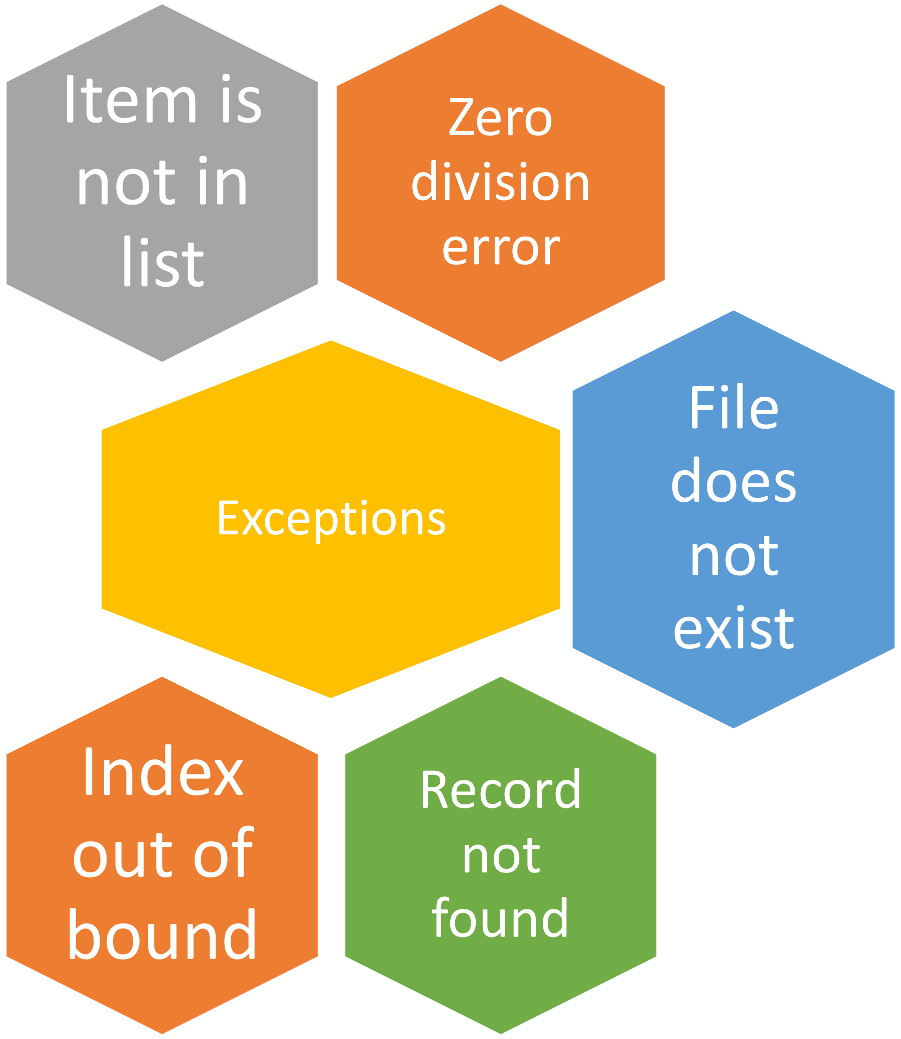
Exceptions and Tools

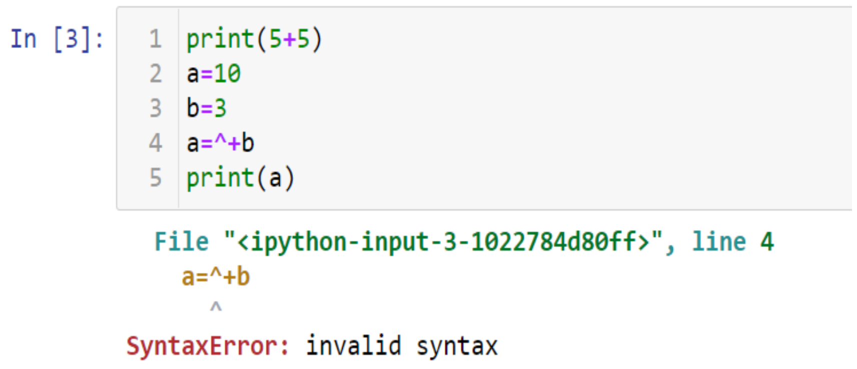
What is an exception

- An unexpected error.
- This type of error occurs whenever syntactically correct Python code results in an error.
- It is different from syntax error.
- Syntax error occur when parser detect an incorrect statement.
- It is not logical error too.

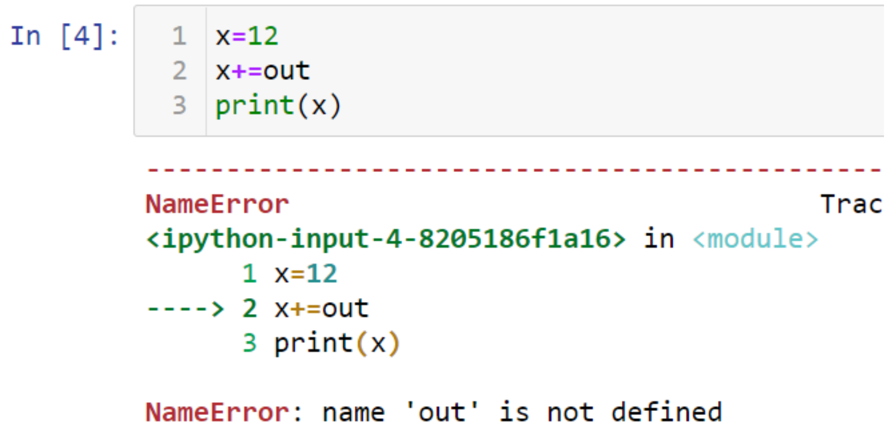


Exception vs Syntax Error

- Syntax error known as parsing error.
- It displays a little ‘arrow’ pointing at the earliest point in the line where the error was detected.
- Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it.
- Errors detected during execution are called *exceptions* and are not unconditionally fatal.
- Python comes with various built-in exceptions as well as the possibility to create user defined exceptions.



```
In [3]:
print(5+5)
a=10
b=3
a=^+b
print(a)

  File "<ipython-input-3-1022784d80ff>", line 4
    a=^+b
      ^
SyntaxError: invalid syntax
```



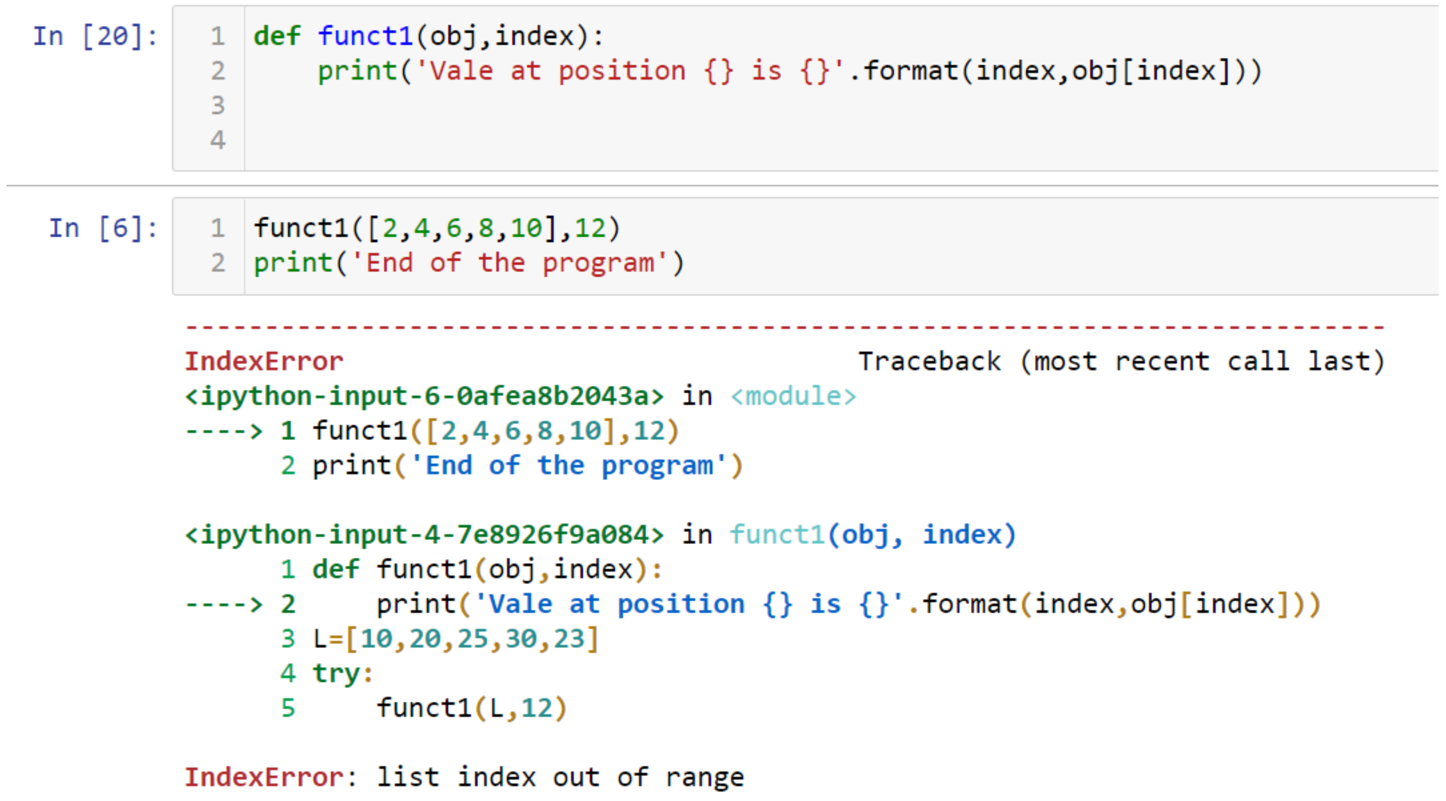
```
In [4]:
x=12
x+=out
print(x)

---------------------------------------------------------------
NameError                                                  Trac
<ipython-input-4-8205186f1a16> in <module>
      1 x=12
----> 2 x+=out
      3 print(x)

NameError: name 'out' is not defined
```

Exception Roles

- Error handling
- Event Notification
- Special-case handling
- Termination actions
- Unusual control flows

Default Exception Handler

- If you don’t catch and handle exceptions then Python’s default exception handler kicks it.



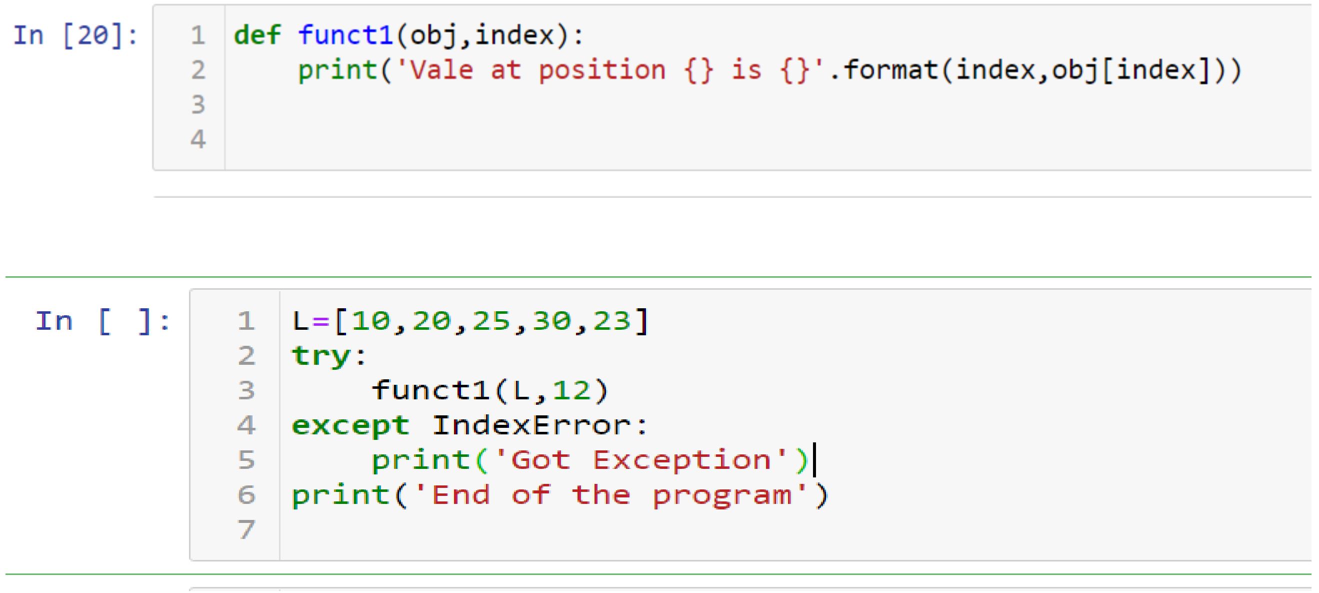
```
In [20]: def funct1(obj,index):
             print('Vale at position {} is {}'.format(index,obj[index]))
```

```
In [6]: funct1([2,4,6,8,10],12)
        print('End of the program')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-0afea8b2043a> in <module>
----> 1 funct1([2,4,6,8,10],12)
      2 print('End of the program')

<ipython-input-4-7e8926f9a084> in funct1(obj, index)
      1 def funct1(obj,index):
----> 2     print('Vale at position {} is {}'.format(index,obj[index]))
      3 L=[10,20,25,30,23]
      4 try:
      5     funct1(L,12)

IndexError: list index out of range
```

Handle Exception

- Server programs, for instance, typically need to remain active even after internal errors.
- If you don’t want the default exception behavior, wrap the call in a try statement to catch exceptions yourself:

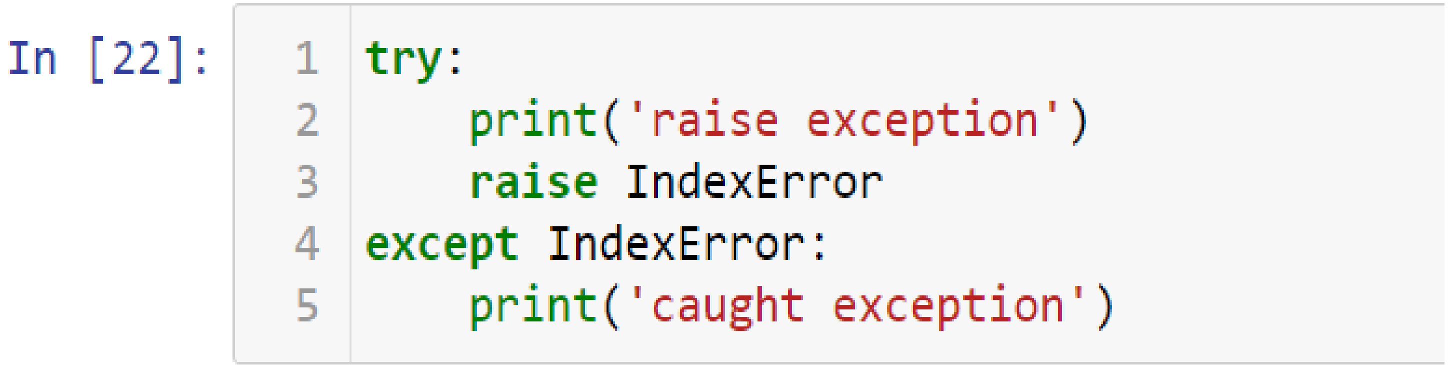


```
In [20]:
def funct1(obj,index):
    print('Vale at position {} is {}'.format(index,obj[index]))
```

```
In [ ]:
L=[10,20,25,30,23]
try:
    funct1(L,12)
except IndexError:
    print('Got Exception')
print('End of the program')
```

Raise exception

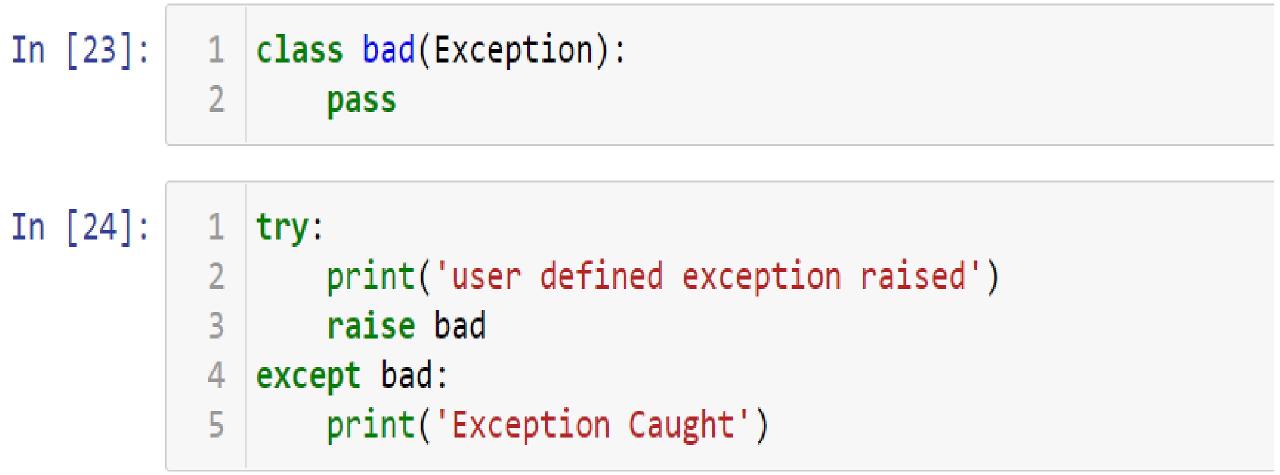
- exceptions can be raised by Python or by your program, and can be caught or not.
- To trigger an exception manually, simply run a raise statement.
- User-triggered exceptions are caught the same way as those Python raises.



```
In [22]:
try:
    print('raise exception')
    raise IndexError
except IndexError:
    print('caught exception')
```

User Defined Exception

- We can define new exceptions of our own that are specific to our programs.
- User-defined exceptions are coded with classes, which inherit from a built-in exception class: usually the class named Exception.

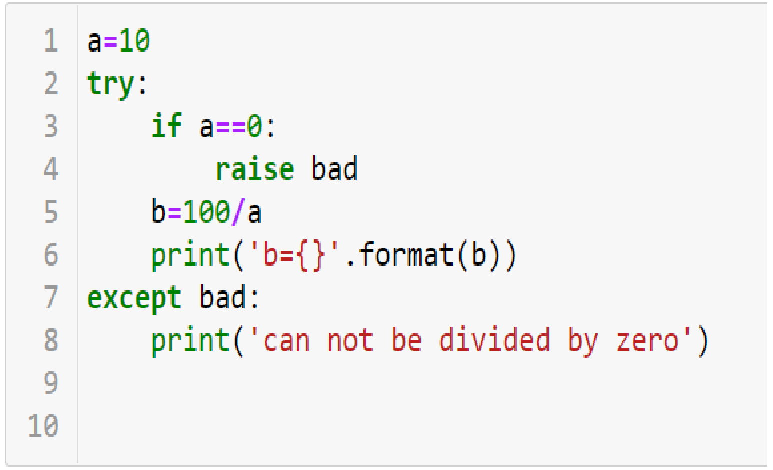


In [23]:

```
class bad(Exception):
    pass
```

In [24]:

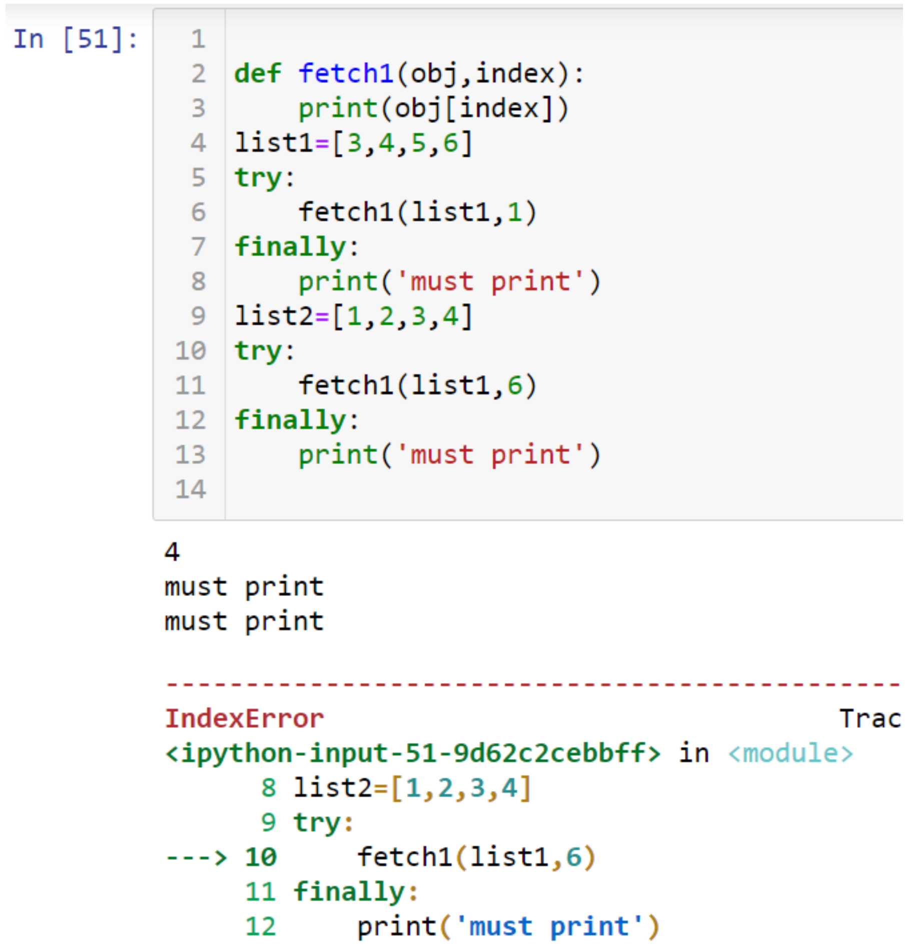
```
try:
    print('user defined exception raised')
    raise bad
except bad:
    print('Exception Caught')
```



```
a=10
try:
    if a==0:
        raise bad
    b=100/a
    print('b={}'.format(b))
except bad:
    print('can not be divided by zero')
```

Termination Actions

- The try/finally combination specifies termination actions that always execute “on the way out,” regardless of whether an exception occurs in the try block.

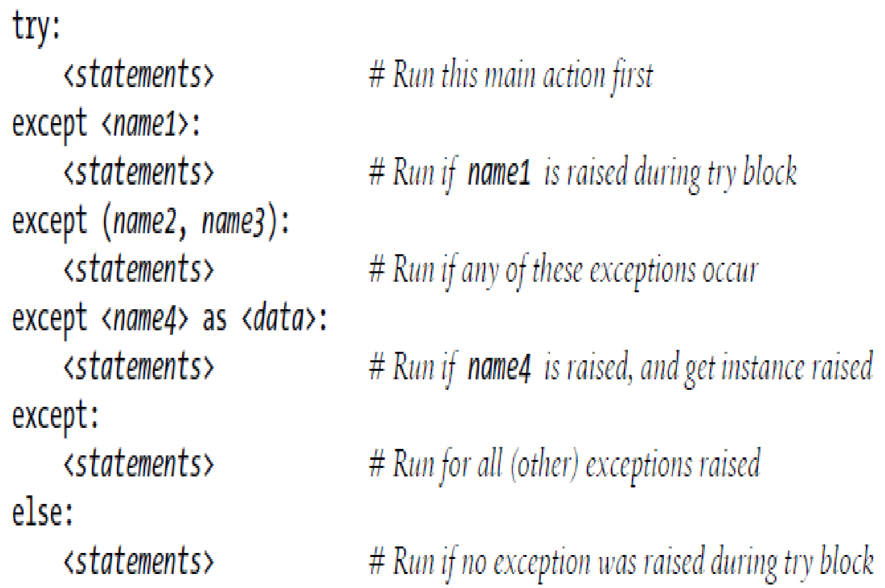


```
In [51]:
def fetch1(obj,index):
    print(obj[index])
list1=[3,4,5,6]
try:
    fetch1(list1,1)
finally:
    print('must print')
list2=[1,2,3,4]
try:
    fetch1(list1,6)
finally:
    print('must print')
```

```
4
must print
must print

---------------------------------------------------------------
IndexError                                                  Trac
<ipython-input-51-9d62c2cebbff> in <module>
      8 list2=[1,2,3,4]
      9 try:
---> 10     fetch1(list1,6)
     11 finally:
     12     print('must print')
```

Try/except/else statement

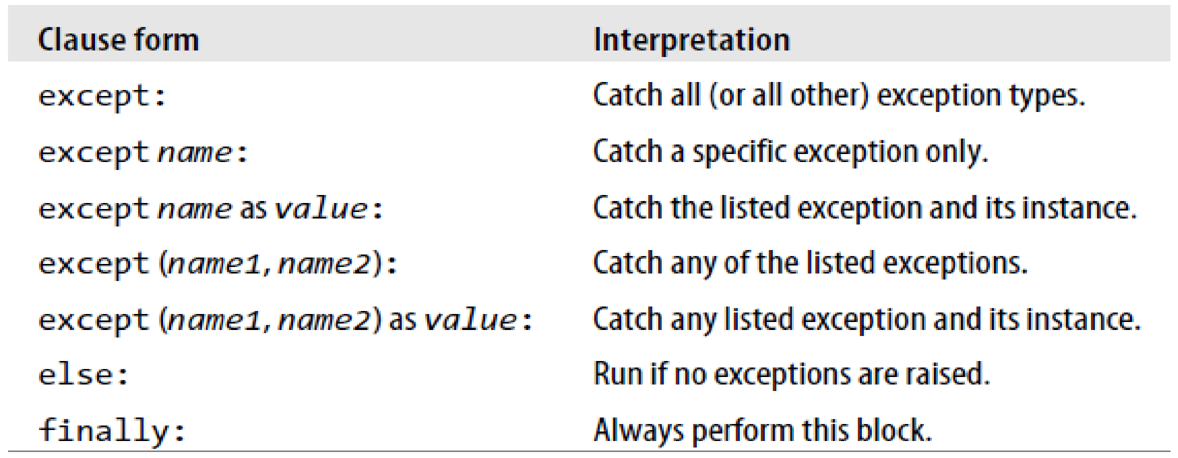
- The try is a compound statement.
- It starts with a try header line, followed by a block of (usually) indented statements, then one or more except clauses that identify exceptions to be caught, and an optional else clause at the end.



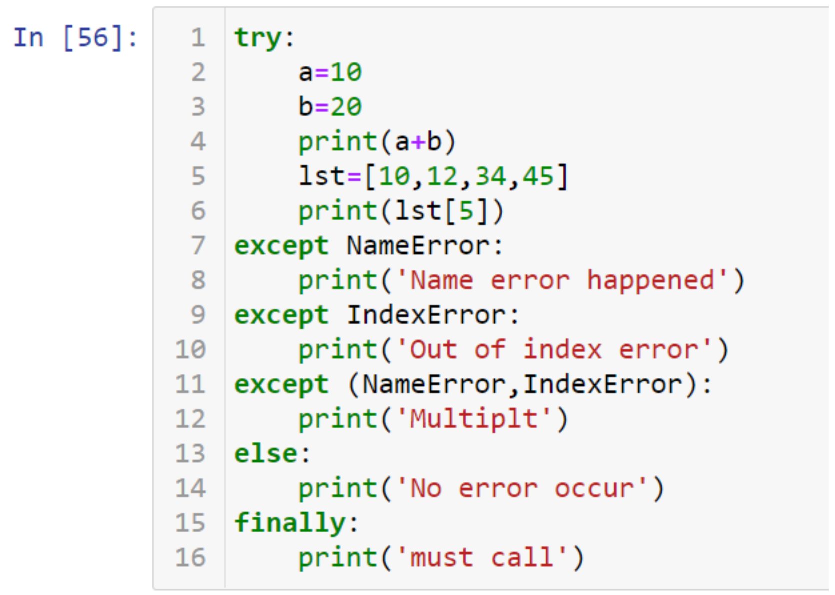
```
try:
    <statements>              # Run this main action first
except <name1>:
    <statements>              # Run if name1 is raised during try block
except (name2, name3):
    <statements>              # Run if any of these exceptions occur
except <name4> as <data>:
    <statements>              # Run if name4 is raised, and get instance raised
except:
    <statements>              # Run for all (other) exceptions raised
else:
    <statements>              # Run if no exception was raised during try block
```

Try statement clause

- a variety of clauses can appear after the try header.
- Must use at least one.



Clause form	Interpretation
except:	Catch all (or all other) exception types.
except *name*:	Catch a specific exception only.
except *name* as *value*:	Catch the listed exception and its instance.
except (*name1*, *name2*):	Catch any of the listed exceptions.
except (*name1*, *name2*) as *value*:	Catch any listed exception and its instance.
else:	Run if no exceptions are raised.
finally:	Always perform this block.



```
In [56]:
try:
    a=10
    b=20
    print(a+b)
    lst=[10,12,34,45]
    print(lst[5])
except NameError:
    print('Name error happened')
except IndexError:
    print('Out of index error')
except (NameError,IndexError):
    print('Multiplt')
else:
    print('No error occur')
finally:
    print('must call')
```

Bibliography

- Realpython.com
- Learning Python book