# 19.1. What is an exception?

An *exception* is a signal that a condition has occurred that can't be easily handled using the normal flow-of-control of a Python program. *Exceptions* are often defined as being "errors" but this is not always the case. All errors in Python are dealt with using *exceptions*, but not all *exceptions* are errors.

# 19.2. Exception Handling Flow-of-control

To explain what an *exception* does, let's review the normal "flow of control" in a Python program. In normal operation Python executes statements sequentially, one after the other. For three constructs, if-statements, loops and function invocations, this sequential execution is interrupted.

- For *if-statements*, only one of several statement blocks is executed and then flow-of-control jumps to the first statement after the if-statement.
- For *loops*, when the end of the loop is reached, flow-of-control jumps back to the start of the loop and a test is used to determine if the loop needs to execute again. If the loop is finished, flow-of-control jumps to the first statement after the loop.
- For *function invocations*, flow-of-control jumps to the first statement in the called function, the function is executed, and the flow-of-control jumps back to the next statement after the function call.

Do you see the pattern? If the flow-of-control is not purely sequential, it always executes the first statement immediately following the altered flow-of-control. That is why we can say that Python flow-of-control is sequential. But there are cases where this sequential flow-of-control does not work well.

Exceptions provide us with way way to have a non-sequential point where we can handle something out of the ordinary (exceptional).

## 19.2.1. Raising and Catching Errors

The try/except control structure provides a way to process a run-time error and continue on with program execution. Until now, any run-time error, such asking for the 8th item in a list with only 3 items, or dividing by 0, has caused the program execution to stop. In the browser ActiveCode windows, you get an error message in a box below. When you are executing python programs from the command-line, you also get an error message saying something about what went wrong and what line it occurred on. After the run-time error is encountered, the python interpreter does not try to execute the rest of the code. You have to make some change in your code and rerun the whole program.

With try/except, you tell the python interpreter:

- **Try to execute a block of code, the "try" clause.**
  - If the whole block of code executes without any run-time errors, just carry on with the rest of the program after the try/except statement.
- **If a run-time error does occur during execution of the block of code:**
  - skip the rest of that block of code (but don't exit the whole program)
  - execute a block of code in the "except" clause
  - then carry on with the rest of the program after the try/except statement

```
try:
   <try clause code block>
except <ErrorType>:
   <exception handler code block>
```

The syntax is fairly straightforward. The only tricky part is that after the word except, there can optionally be a specification of the kinds of errors that will be handled. The catchall is the class Exception. If you write `except Exception:` all runtime errors will be handled. If you specify a more restricted class of errors, only those errors will be handled; any other kind of error will still cause the program to stop running and an error message to be printed.

The code below causes an error of type IndexError, by trying to access the third element of a two-element list.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
1 items = ['a', 'b']
2 third = items[2]
3
```

Activity: 1 -- ActiveCode (exceptions_1)

### Error

```
IndexError: list index out of range on line 2
```

### Description

This message means that you are trying to index past the end of a string or a list. For example if your list has 3 things in it and you try to access the item at position 3 or more.

### To Fix

Remember that the first item in a list or string is at index position 0, quite often this message comes about because you are off by one. Remember in a list of length 3 the last legal index is 2

The code below causes an error of type ZeroDivisionError, or less specifically ArithmeticError.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
x = 5
y = x/0

```

Activity: 2 -- ActiveCode (exceptions_2)

### Error

```
ZeroDivisionError: integer division or modulo by zero on line 2
```

### Description

This tells you that you are trying to divide by 0. Typically this is because the value of the variable in the denominator of a division expression has the value 0

### To Fix

You may need to protect against dividing by 0 with an if statment, or you may need to rexamine your assumptions about the legal values of variables, it could be an earlier statment that is unexpectedly assigning a value of zero to the variable in question.

Let's see what happens if we wrap some of this problematic code in a try/except statement. Note that `this won't print` doesn't print: when the error is encountered, the rest of the try block is skipped and the exception block is executed. When the except block is done, it continues on with the next line of code that's outdented to the same level as the try: `continuing` is printed.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
try:
    items = ['a', 'b']
    third = items[2]
    print("This won't print")
except Exception:
    print("got an error")

print("continuing")

```

```
got an error
continuing
```

Activity: 3 -- ActiveCode (exceptions_3)

If we catch only IndexEror, and we actually have a divide by zero error, the program does stop executing.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
try:
    items = ['a', 'b']
    third = items[2]
    print("This won't print")
except IndexError:
    print("error 1")
```

```
print("continuing")

try:
    x = 5
    y = x/0
    print("This won't print, either")
except IndexError:
    print("error 2")
```

```
error 1
continuing
```

Activity: 4 -- ActiveCode (exceptions_4)

## Error

```
ZeroDivisionError: integer division or modulo by zero on line 12
```

## Description

This tells you that you are trying to divide by 0. Typically this is because the value of the variable in the denominator of a division expression has the value 0

## To Fix

You may need to protect against dividing by 0 with an if statment, or you may need to rexamine your assumptions about the legal values of variables, it could be an earlier statment that is unexpectedly assigning a value of zero to the variable in question.

There's one other useful feature. The exception code can access a variable that contains information about exactly what the error was. Thus, for example, in the except clause you could print out the information that would normally be printed as an error message but continue on with execution of the rest of the program. To do that, you specify a variable name after the exception class that's being handled. The exception clause code can refer to that variable name.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
try:
    items = ['a', 'b']
    third = items[2]
    print("This won't print")
except Exception as e:
    print("got an error")
    print(e)

print("continuing")

```

```
got an error
IndexError: list index out of range on line 3
continuing
```

Activity: 5 -- ActiveCode (exceptions_5)

**Check your understanding**

exceptions-1: Which type of error can be noticed and handled using try/except?

- A. syntax
- B. run-time
- C. semantic

Check me | Compare me

✔ Run-time errors like index out of bounds can be caught and handled gracefully with try/except.

Activity: 6 -- Multiple Choice (exceptions_mc_1)

exceptions-2: When a run-time exception of type ZeroDivisionError occurs, and you have a statement `except IndexError`, the program will stop executing completely.

- A. True
- B. False

Check me | Compare me

✔ If your code is only catching IndexError errors, then the exception will not be handled, and execution will terminate.

Activity: 7 -- Multiple Choice (exceptions_mc_2)

exceptions-3: After a run-time exception is handled by an except clause, the rest of the code in the try clause will be executed.

- A. True
- B. False

Check me | Compare me

✔ The rest of the code after the whole try/except statement will execute, but not the rest of the code in the try block.

Activity: 8 -- Multiple Choice (exceptions_mc_3)

exceptions-4: How many lines will print out when the following code is executed?

```
try:
    for i in range(5):
        print(1.0 / (3-i))
except Exception as error_inst:
    print("Got an error", error_inst)
```

- A. 0
- B. 1
- C. 3
- D. 4
- E. 5

Check me | Compare me

✔ It will print the fraction for three values of i, and then one error message.

Activity: 9 -- Multiple Choice (exceptions_mc_4)

5. Below, we have provided a list of tuples that consist of student names, final exam scores, and whether or not they will pass the class. For some students, the tuple does not have a third element because it is unknown whether or not they will pass. Currently, the for loop does not work. Add a try/except clause so the code runs without an error - if there is no third element in the tuple, no changes should be made to the dictionary.

Save & Run | 5/15/2021, 2:35:56 PM - 4 of 4 | Show in CodeLens

```
students = [('Timmy', 95, 'Will pass'), ('Martha', 70), ('Betty', 82, 'Will pass'),
passing = {'Will pass': 0, 'Will not pass': 0}
for tup in students:
    try:
        if tup[2]=='Will pass':
            passing['Will pass'] += 1
        elif tup[2]=='Will not pass':
            passing['Will not pass'] += 1
    except:
        pass
```

Activity: 10 -- ActiveCode (ee_exceptions_012)

| Result | Actual Value | Expected Value | Notes |
|---|---|---|---|
| Pass | {'Wil...': 2} | {'Wil...': 2} | Testing that passing is created correctly. |

Expand Differences

You passed: 100.0% of the tests

6. Below, we have provided code that does not run. Add a try/except clause so the code runs without errors. If an element is not able to undergo the addition operation, the string 'Error' should be appended to plus_four.

Save & Run | 5/15/2021, 2:57:07 PM - 23 of 23 | Show in CodeLens

```
nums = [5, 9, '4', 3, 2, 1, 6, 5, '7', 4, 3, 2, 6, 7, 8, '0', 3, 4, 0, 6, 5, '3',

plus_four = []

for num in nums:
    try:
        plus_four.append(num+4)
    except:
        if num in nums == ('4','7','0','3','1','9'):
            replace(num,'Error')
        else:
            pass
```

Activity: 11 -- ActiveCode (ee_exceptions_022)

| Result | Actual Value | Expected Value | Notes |
|---|---|---|---|
| Fail | [9, 1...9, 5] | [9, 1...9, 5] | Testing that plus_four is created correctly. |

Expand Differences

You passed: 0.0% of the tests

You have attempted 12 of 11 activities on this page

19. Exceptions">

19.3. When to use try/except">

✔ Completed. Well Done!

19.3. When to use try/except">Next Section - 19.3. When to use try/except



| Back to top