הטכניון
היחידה ללימודי המשך
ולימודי חוץ עייש עזריאלי

Error handling

TECHNION
Azrieli Continuing Education and
External Studies Division

1. What are Exceptions.
2. Raise
3. Handling exceptions
4. Program flow

Exception

```
1/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Tra
<ipython-input-8-9e1622b385b6> in <module>()
----> 1 1/0

ZeroDivisionError: division by zero
```

```
print(name)
```

```
---------------------------------------------------------------------------
NameError                                 Tr
<ipython-input-9-9ba126b17b03> in <module>()
----> 1 print(name)

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

```
a = []
a[0]
```

```
---------------------------------------------------------------------------
IndexError                                Trace
<ipython-input-10-52c7b74b8e70> in <module>()
      1 a = []
----> 2 a[0]

IndexError: list index out of range
```

Raise

- You can throw exceptions by yourself using ‘raise’:

```python
def divide(num1, num2):
  if num2 == 0:
    raise ZeroDivisionError("The second number cannot be 0")
  return num1 / num2

divide(1, 0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
<ipython-input-18-2d2e2aff2fdb> in <module>()
      4   return num1 / num2
      5
----> 6 divide(1, 0)

<ipython-input-18-2d2e2aff2fdb> in divide(num1, num2)
      1 def divide(num1, num2):
      2   if num2 == 0:
----> 3     raise ZeroDivisionError("The second number cannot be 0")
      4   return num1 / num2
      5

ZeroDivisionError: The second number cannot be 0
```

Raise

- Also useful when you don't know how to handle the error and want to pass it over to the calling function.

```python
def divide(num1, num2):
  try:
    return num1 / num2
  except ZeroDivisionError:
    print(f'Error when dividing number {num1} by number {num2}')
    raise

print(f'No error: {divide(1, 2)}')

print('With error:')
try:
  divide(1, 0)
except Exception as e:
  print(e)
```

```
No error: 0.5
With error:
Error when dividing number 1 by number 0
division by zero
```

Handle exception

```
try:
    # do something that might fail
except Exception_Type:
    # do something in case of error
```

- Wrap your risky code with try/except closures:

```
try:
  1/0
except ZeroDivisionError:
  print('An error cought!')
```

```
An error cought!
```

Handle exception

```
try:
  1/0
except ZeroDivisionError as e:
  print(f'An error cought! Details: {e}')
```

```
An error cought! Details: division by zero
```

- Usually, we don’t want the program to stop running without knowing why.
- Use the except closure to print details about the error:

Handle exception

- Chaining exception handling:

```
try:
    # Do something that might fail
    file.write()
except PermissionError:
    # If we don't have permission to do the operation (e.g. write to protected disk), do the following
    # ...
except IsADirectoryError:
    # Trying to do a file operation on a directory - so do the following
    # ...
except (NameError, TypeError):
    # If we encouter either a non-existent variable or operation on variables, do the following
    # ...
except Exception:
    # General error, not caught by previous exceptions
    # ...
```

Generic exception

Handle exception

- Other useful exception chaining: else, finally

```
try:
    # Do something that might fail
    file.write()
except PermissionError:
    # If we don't have permission to do the operation (e.g. write to protected disk), do the following
    # ...
except IsADirectoryError:
    # Trying to do a file operation on a directory - so do the following
    # ...
except (NameError, TypeError):
    # If we encouter either a non-existent variable or operation on variables, do the following
    # ...
except Exception:
    # General error, not caught by previous exceptions
    # ...
else:
    # If the operation under "try" succeeded, do the following
    # ...
finally:
    # Regardless of the result - success or failure - do this.
    # ...
```

Handle exception

- Else, finally example:

```python
def divisor(a, b):
    try:
        ans = a / b
    except ZeroDivisionError as e:
        ans = None
        err = e
    else:
        err = None
    finally:
      return ans, err

# Should work:
ans, err = divisor(1, 2)
print(ans, " ----", err)

# ZeroDivisionError:
ans, err = divisor(1, 0)
print(ans, "----", err)
```

```
0.5  ---- None
None ---- division by zero
```

Summary

Exception

- Error handling, chaining

Raise

- Move error to higher function

Thank you