Object Oriented Programming with Python

Gramsci Hermozo

Session 06

Content

- The Clock
- Exceptions
- Error Handling
- Manage Errors friendly
- Logs

Exe. Complete the clock

We want to build a Clock with a European style (24 hours). Where the display shows the time from *00:00* (midnight) to *23:59* (one minute before midnight)

Exceptions

Errors detected during execution are called exception and are not unconditionally fatal.

```
"""
Most common exception handlend in python
"""
Exception # Base Exception
AttributeError
ZeroDivisionError
ArithmeticError
IndexError
SyntaxError
TypeError
...
```

Error Handling

Use `try...except` to catch exceptions.

```
try:
  x = 5/0
except ZeroDivisionError as ex:
  print("Division by zero!!!! The exception: {}".format(e)
```

Error Handling

```
# Don't catch everything
try:
  function_trow_except()
except Exception as ex:
  # TODO: do something with the error
```

Error Handling

```
# Caching multiple Exception
try:
  d = {}
  a = d[1]
  b = d.no_implemented_function()
except (KeyError, AttributeError) as e:
  print("A keyerror or an AttributeError exception")
```

Error Handling

```
# Separate exception block for each type.
try:
  a = {}
  b = a[1]
  d = b.no_implemented_funciton()
except KeyError as e:
  print("This is a keyerror. Exception message:", e)
except AttributeError as ex:
  print("Attribute error. Exception message:", ex)
```

Error Handling

```
# Else and finally
try:
  # Exception raise
except Exception as ex:
  pass
else:
  # This section would be executed just if no exception is
finally:
  # This section of code would be executed always even wit
```

Raise Excepton

```
def function_rise_except():
  raise ValueError("This function is throwing an exception"
```

Logs

```
# use logging package
import logging

# Configure logging
logging.basicConfig(filename='test.log', level=logging.DEBU

# how to use
logging.debug('<Class_name>:<Function_name>: {}'.format(arg
```