Lab -5

Programming in Python

Instructor : AALWAHAB DHULFIQAR

Advisor : Dr. Tejfel Mate

python

What you will learn:

More about exceptions.
os module
Introduction to the datetime.
What is Raspberry Pi .
Assignments

Use your brain to find the answers

Assume that the following piece of code has been successfully executed:

```
class Dog:
    kennel = 0
    def __init__(self, breed):
        self.breed = breed
        Dog.kennel += 1
    def __str__(self):
        return self.breed + " says: Woof!"
class SheepDog(Dog):
    def __str__(self):
        return super().__str__() + " Don't run away, Little Lamb!"

class GuardDog(Dog):
    def __str__(self):
        return super().__str__() + " Stay where you are, Mister Intruder!"

rocky = SheepDog("Collie")
luna = GuardDog("Dobermann")
```

print(rocky) print(luna)
Collie says: Woof! Don't run away, Little Lamb! Dobermann says: Woof! Stay where you are, Mister Intruder!
print(issubclass(SheepDog, Dog), issubclass(SheepDog, GuardDog)) print(isinstance(rocky, GuardDog), isinstance(luna, GuardDog))
print(luna is luna, rocky is luna) print(rocky.kennel)

More about exceptions

```
def reciprocal(n):
    try:
        n = 1 / n
    except ZeroDivisionError:
        print("Division failed")
        n = None
    else:
        print("Everything went fine")
    finally:
        print("It's time to say goodbye")
        return n
print(reciprocal(2))
print(reciprocal(0))
```

```
try:
    i = int("Hello!")
except Exception as e:
    print(e)
    print(e.__str__())
```

Exceptions are classes

```
def print_exception_tree(thisclass, nest = 0):
    if nest > 1:
        print("    |" * (nest - 1), end="")
    if nest > 0:
        print("    +---", end="")

    print(thisclass.__name__)

    for subclass in thisclass.__subclasses__():
        print_exception_tree(subclass, nest + 1)


print_exception_tree(BaseException)
```

```
class MyZeroDivisionError(ZeroDivisionError):

    pass


def do_the_division(mine):
    if mine:
        raise MyZeroDivisionError("some worse
news")
    else:
        raise ZeroDivisionError("some bad
news")


for mode in [False, True]:
    try:
        do_the_division(mode)
    except ZeroDivisionError:
        print('Division by zero')

for mode in [False, True]:
    try:
        do_the_division(mode)
    except MyZeroDivisionError:
        print('My division by zero')
    except ZeroDivisionError:
        print('Original division by zero')
```

Lab 6.1

Scenario

Prof. Jekyll conducts classes with students and regularly makes notes in a text file. Each line of the file contains three elements: the student's first name, the student's last name, and the number of point the student received during certain classes.

The elements are separated with white spaces. Each student may appear more than once inside Prof. Jekyll's file.

The file may look as follows:

```
John    Smith    5
Anna    Boleyn   4.5
John    Smith    2
Anna    Boleyn   11
Andrew  Cox      1.5
```

```
class StudentsDataException(Exception):
    pass


class BadLine(StudentsDataException):
    # Write your code here.


class FileEmpty(StudentsDataException):
    # Write your code here.
```

```
Andrew Cox      1.5
Anna Boleyn     15.5
John Smith      7.0
```

os module

lets you interact with the operating system using Python.

```
import os
print(os.uname())
```

Creating directories in Python

```
import os

os.mkdir("my_first_directory")
print(os.listdir())
```

Recursive directory creation

```
import os

os.makedirs("my_first_directory/my_second_directory")
os.chdir("my_first_directory")
print(os.listdir())
```

Where am I now?

```
import os
os.makedirs("my_first_directory/my_second_directory")
os.chdir("my_first_directory")
print(os.getcwd())
os.chdir("my_second_directory")
print(os.getcwd())
```

Deleting directories in Python

```
import os
os.mkdir("my_first_directory")
print(os.listdir())
os.rmdir("my_first_directory")
print(os.listdir())
```

The system() function

All functions presented in this part of the course can be replaced by a function called system, which executes a command passed to it as a string.

```
import os
returned_value = os.system("mkdir my_first_directory")
print(returned_value)
```

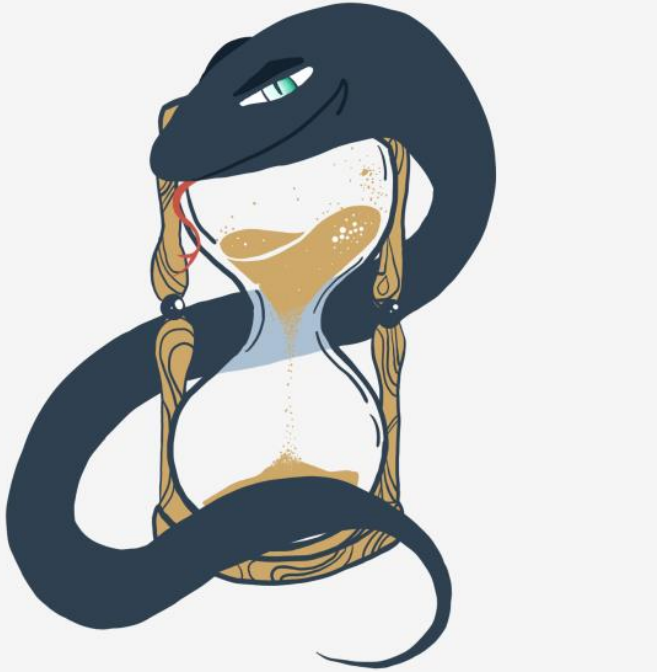
Introduction to the `datetime` module

Getting the current local date and creating date objects

```
from datetime import date

today = date.today()

print("Today:", today)
print("Year:", today.year)
print("Month:", today.month)
print("Day:", today.day)
```

Creating a date object from a timestamp

```
from datetime import date
import time

timestamp = time.time()
print("Timestamp:", timestamp)

d = date.fromtimestamp(timestamp)
print("Date:", d)
```



Creating a date object using the ISO format

```
from datetime import date

d = date.fromisoformat('2019-11-04')
print(d)
```

What day of the week is it?

```
from datetime
import dated = date(2022,3, 13)
print(d.weekday())
```

Creating `time` objects

```
from datetime import time
t = time(14, 53, 20, 1)
print("Time:", t)
print("Hour:", t.hour)
print("Minute:", t.minute)
print("Second:", t.second)
print("Microsecond:",
t.microsecond)
```

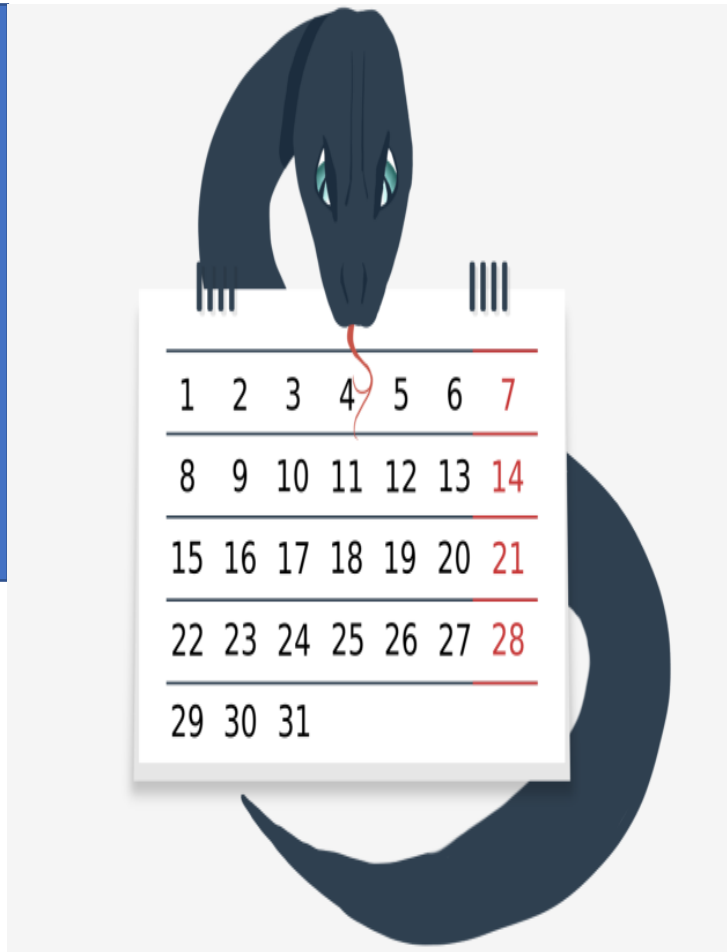
Getting a timestamp

The timestamp method returns a float value expressing the number of seconds elapsed between the date and time indicated by the datetime object and January 1, 1970, 00:00:00 (UTC).

```
from datetime import datetime

dt = datetime(2020, 10, 4, 14, 55)
print("Timestamp:", dt.timestamp())
```

Date and time formatting

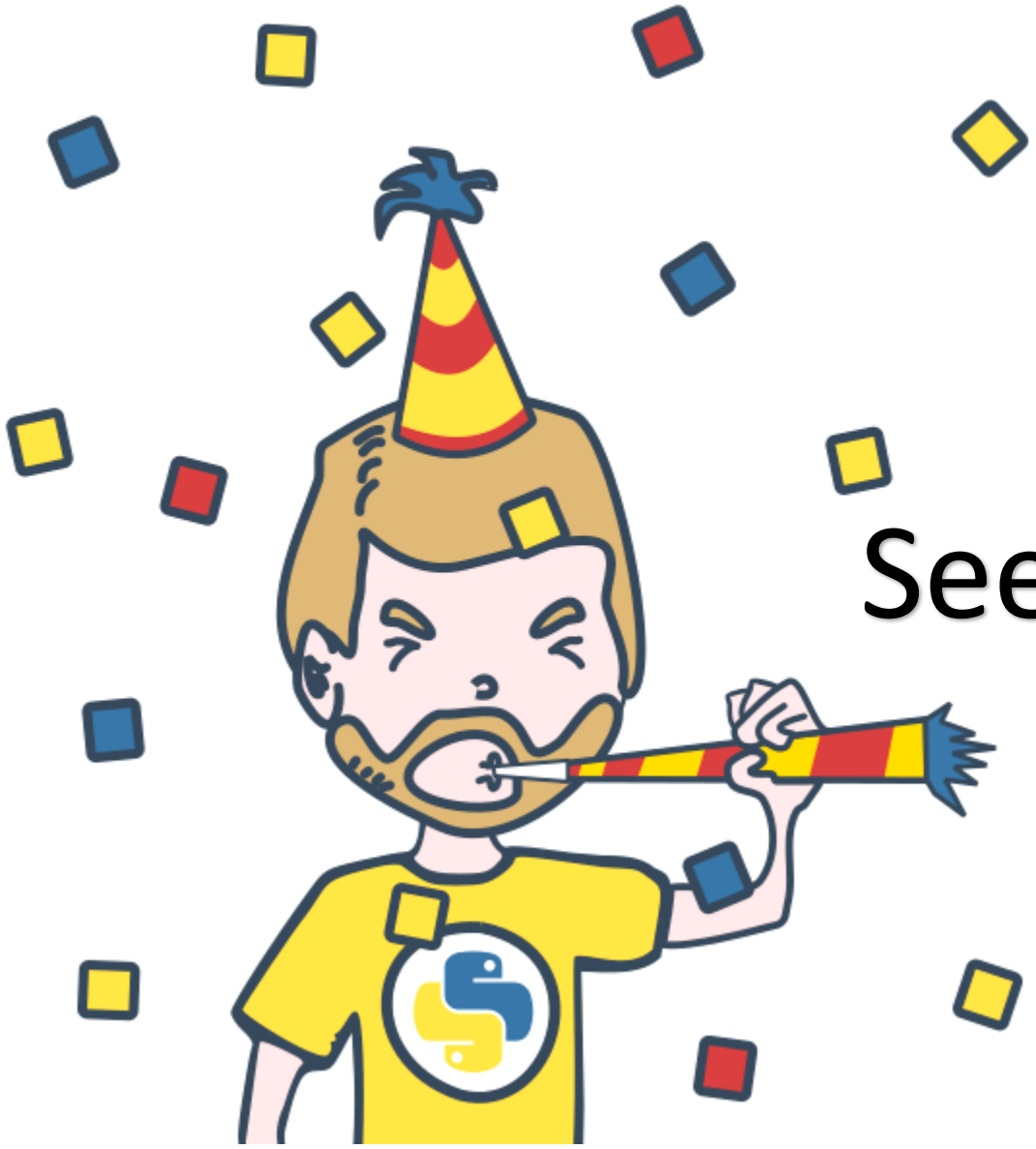
```
from datetime import time
from datetime import datetime

t = time(14, 53)
print(t.strftime("%H:%M:%S"))

dt = datetime(2020, 11, 4, 14, 53)
print(dt.strftime("%y/%B/%d %H:%M:%S"))
```




```
import calendar
print(calendar.calendar(2020))
```

```
import calendar
print(calendar.month(2020, 11))
```

```
import calendar

calendar.setfirstweekday(calendar.SUNDAY)
calendar.prmonth(2020, 12)
```



See you Next week ☺

Assignment