Formal Languages and Compilers

Lab3

Python
-file handling-

- *open(filename, mode)* –function that allow us to work with files (it opens them for python)
- mode: (you can specify if a file should be handled as text (t)-default- or as binary (b)- e.g., images)
  - r (read)-default; if the file does not exist=>error
  - a (append) if the file does not exist => it creates it
  - w (write) if the file does not exist => it creates it
  - x (create) if the file exists => error
- *close()* –close the file
- *read()* - method for reading a file; readline()-method for reading only the first line
- *write()* – method for adding(mode a)/overwriting(mode w) file
- Delete file – os module: *os.remove()* (*os.rmdir()* to delete a folder)

Python
-file handling-

- Simple example

filehandling.txt - Notepad

File Edit Format View Help

```
Hello!
This is an example for file handling.
Have a great day!
Best regards
```

filehandling2.txt - Notepad

File Edit Format View Help

```
Hello, user!
This is an example for file handling.
Have a great day!
Best regards
```

```
#read the whole file
fh = open("filehandling.txt")
print(fh.read())

print("----")
#reads first 6 characters
fh2 = open("filehandling2.txt")
print(fh2.read(12))

print("----")

fh3 = open("filehandling2.txt", "a")
fh3.write("\nI have added a new line!")
fh3.close()

#open and read the file after the appending:
fh4 = open("filehandling2.txt", "r")
print(fh4.read())

print("----")

fh5 = open("filehandling.txt", "w")
fh5.write("Woops! I have deleted the initial text and replaced it with the new one!")
fh5.close()

#open and read the file after the appending:
fh5 = open("filehandling.txt", "r")
print(fh5.read())
```

filehandling.txt - Notepad

File Edit Format View Help

```
Woops! I have deleted the initial text and replaced it with the new one!
```

filehandling2.txt - Notepad

File Edit Format View Help

```
Hello, user!
This is an example for file handling.
Have a great day!
Best regards


I have added a new line!
```

```
Hello!
This is an example for file handling.
Have a great day!
Best regards
----
Hello, user!
----
Hello, user!
This is an example for file handling.
Have a great day!
Best regards


I have added a new line!
----
Woops! I have deleted the initial text and replaced it with the new
one!
```

Python
-file handling-

Method	Description
close()	Close the file
read()	Returns the file content
readable()	Returns whether or not the file can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change file position
seekable()	Returns whether or not the file can have its position changed
tell()	Returns the current file position
truncate()	Resizes the file to a specified size
writeable()	Returns whether or not the file can be written to or not
write()	Writes the specified string to the filte
writelines()	Writes a list of strings to the file

Python
-Modules-

- **Module**: file which contains functions that you want to use in your application
- You can create your own modules, or you can import build-in modules
- Creating a module: save the code you want to use in a .py file
- **Using a module**: in the file you want to use the module write *import moduleName* (make sure you save the module and the file where you want to use it in the same folder)
- You can rename how you want to use the module by *writing import moduleName as alias*
- You can also import only parts of the module by writing *from moduleName import part* (when you do this, you don't need to write the moduleName when referring to elements from it)

Python
-Modules: creating-

- Simple example

module.py file

```
def hello(name):
    print ("Hello, ",name)
```

file where I use the module

```
import module

module.hello("Alle")
```

```
import module as m

m.hello("Alle")
```

Output

```
Hello,  Alle
```

```
myDict={"name": "Perfume: The Story of a Murderer",
  "author": "Patrick Suskind",
  "year": 1985}
```

```
from module import myDict

print(myDict["author"])
```

```
Patrick Suskind
```

Python
-Modules: Built-in-

- There are a lot of build-in modules
- To use a built-in module, you just need to import it: *import moduleName*
- Some of the most used ones are:
  - math (common mathematic functions such as sqrt(), pi, log())
  - os (perform tasks on OS e.g., mkdir(), rmdir())
  - random (functions handling randomization e.g., random() for random numbers)
  - time (time related functions e.g., time(), localtime())

Python
-Modules: Built-in-

- Simple examples

```python
import math

a=25
b=math.sqrt(a)
x= math.pi

print(b)
print("\n",x)


print("----")


import time

print(time.localtime())


print("----")

import random as r

a=r.random()

print(a)


print("----")
```

```
5.0

 3.141592653589793
----
time.struct_time(tm_year=2021, tm_mon=10, tm_mday=3, tm_hour=16,
tm_min=51, tm_sec=51, tm_wday=6, tm_yday=276, tm_isdst=1)
----
0.8437466214169552
```

Python
-Packages-

- Package: collection of modules
- pip: package manager for Python
- Information about packages can be found at https://pypi.org/
- Check if pip is installed: in kernel, type "*pip --version*" (if it is not installed, it can be downloaded from here
- Install packages: in kernel type "*pip install packageName*"
- Remove packages: in kernel type "*pip uninstall packageName*"
- List packages: in kernel type "*pip list*"
- Use package: in code, type "*import packageName*"
- Use modules/sub-packages within packages: in code, type "*from packageName import moduleName*"

Python -datetime()-

- Module with classes helping as work with dates and time
- Formats:

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12

%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00

Python
-datetime()-

- The datetime() class inside the module allows us to create a Date object
- strftime() method is used to format the Date objects
- Example

```
import datetime

x=datetime.datetime(2022,10,25)

print(x)

print(x.strftime('%b'))
```

```
2022-10-25 00:00:00
Oct
```

Exercises

1. Write a program that displays the current date and time with the complete name of the month. (Hint: use datetime.now())
2. Write a program that takes as user input the radius of a circle (as an integer) and displays the area of that circle. $A = \pi r^2$
3. Select a random number from a list. (Hint: use random.choice())
4. Shuffle the elements from a given list. (Hint: use random.shuffle())
5. Use the math module to compute the least common multiple and the greatest common divisor of two numbers.
6. Use the math module to compute the factorial of a number inputted by the user.
7. Write a program that displays the week number of a date. (Hint: use isocalendar())
8. Write a program that computes the distance between two points.

$$\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Homework

1. Create your own module with any function you want and use it in another file.
2. Create a list populated with 5 random numbers (Hint: use randint())
3. Write a function that prints a list of 5 random integers between 40 and 70.
4. Use the datetime module to create a datetime object and print the full name of the weekday of that day.
5. Create a directory. Create a.txt file in it. Add some text to it. Read the first 2 lines. Overwrite the text inside the file.
6. Print the name of the operating system. List the files and directories in the current directory.
7. Write a program that displays the date that was 10days before the current date.