

# ToDos

- <https://www.programiz.com/python-programming/file-operation>

## Links

- ....

```
%%html
<style>
.not_clean {
    background: lightgreen;
    padding: 20px;
}

.comment {
    background: #F3FF70;
    padding: 20px;
}
</style>
<IPython.core.display.HTML object>
```

Današnja naloga bo neka preprosta analiza potnikov na Titaniku.

Podatke bomo uvozili v naš program, jih grafično prikazali in opravili kratko analizo.

## Delo z datotekami

Datoteke uporabljamo, da v njih trajno shranimo podatke.

V splošnem delo z datotekami poteka na sledeč način:

- Odpremo datoteko
- Izvedemo operacijo (pisanje podatkov v datoteko, branje podatkov, itd..)
- Zapremo datoteko (ter tako sprostimo vire, ki so vezani na upravljanje z datoteko -> spomin, procesorska moč, itd..)

### Odpiranje datotek

Python ima že vgrajeno funkcijo `open()` za odpiranje datotek.

Funkcija nam vrne `file object`, imenovan tudi **handle**, s katerim lahko izvajamo operacije nad datoteko.

```
f = open("test.txt")    # open file in current directory
#f = open("C:/Python33/README.txt") # specifying full path
```

Dodatno lahko specificiramo v kakšnem načinu želimo odpreti datoteko.

Lahko jo odpremo v **text mode**. Ko beremo podatke v tem načinu, dobivamo *strings*. To je *default mode*. Lahko pa datoteko odpremo v **binary mode**, kjer podatke beremo kot *bytes*. Takšen način se uporablja pri branju non-text datotek, kot so slike, itd..

Datoteke lahko odpremo v načinu:

- **r** - Podatke lahko samo beremo. (default način)
- **w** - Podatke lahko pišemo v datoteko. Če datoteka ne obstaja jo ustvarimo. Če datoteka obstaja jo prepišemo (če so bli noter podatki jih izgubimo)
- **x** - Ustvarimo datoteko. Če datoteka že obstaja operacija fail-a
- **a** - Odpremo datoteko z namenom dodajanja novih podatkov. Če datoteka ne obstaja jo ustvarimo.
- **t** - odpremo v "text mode" (default mode)
- **b** - odpremo v "binary mode"

```
f = open("test.txt")    # equivalent to 'r' or 'rt'
f = open("test.txt", 'r') # write in text mode
print(type(f))
print(f)

<class '_io.TextIOWrapper'>
<_io.TextIOWrapper name='test.txt' mode='r' encoding='UTF-8'>
f = open("test.txt", 'w') # write in text mode
```

Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).

Moreover, the default encoding is platform dependent. In windows, it is 'cp1252' but 'utf-8' in Linux.

So, we must not also rely on the default encoding or else our code will behave differently in different platforms.

Hence, when working with files in text mode, it is highly recommended to specify the encoding type.

```
f = open("test.txt", mode = 'r', encoding = 'utf-8')
```

## Zapiranje datotek

Ko končamo z našo operacijo moramo datoteko zapreti, ker tako sprostimo vire, ki so vezani na uporabo datoteke (spomin, procesorska moč, itd..).

```
f = open("test.txt", "a")
# perform file operations

f.close()
```

na Linuxu ne dela ta `f.close()`. Še kr lah spreminjam. Ko se python zapre bo počistu. Če pa maš program k laufa dolgo, potem je problem.

Tak način upravljanja z datotekami ni najbolj varen. Če smo odprli datoteko in potem med izvajanjem operacije nad datoteko pride do napake, datoteke ne bomo zaprli.

Varnejši način bi bil z uporabo **try-finally**.

```
try:
    f = open("test.txt", "a")
    # perform file operations
    raise ValueError
finally:
    f.close()
```

*# če ta koda deluje, potem bi morali biti zmožni spreminjati datoteko tudi potem, ko dobimo error.*

```
-----
-----
ValueError                                Traceback (most recent call
last)
c:\Users\nkast\ICTA\python-osnovni-public\Termin06\
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb Cell 24 line
4
      <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X32sZmlsZQ
%3D%3D?line=1'>2</a>      f = open("test.txt", "a")
      <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X32sZmlsZQ
%3D%3D?line=2'>3</a>      # perform file operations
----> <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X32sZmlsZQ
%3D%3D?line=3'>4</a>      raise ValueError
      <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X32sZmlsZQ
```

```
%3D%3D?line=4'>5</a> finally:
    <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X32sZmlsZQ
%3D%3D?line=5'>6</a>      f.close()
```

ValueError:

### Python with Context Managers

Isto stvar dosežemo z uporabo `with` statement.

```
with open("test.txt", "a") as f:
    pass
    # perform file operations
print(f.closed)
True
```

dodat zadevo glede fopen flag da vidjo kdaj je zaprt

## Branje datotek

Za branje, datoteko odpremo v `read(r)` načinu.

(Imamo datoteko katere vsebina je: *Hello World!*  
*This is my file.*)

```
with open("test.txt", 'r') as f:
    file_data = f.read()    # read all data
    print(file_data)

#print(file_data)
#file_data

with open("test.txt", "r") as f:
    file_data = f.read(2) # read the first 2 data
```

```

print(file_data)
file_data = f.read(6) # read the next 6 data
print(file_data)
file_data = f.read() # reads till the end of the file
print(file_data)
file_data = f.read() # further reading returns empty string
print(file_data)

```

We can see that, the read() method returns newline as '\n'. Once the end of file is reached, we get empty string on further reading.

Po datoteki se lahko tudi premikamo z uporabo seek() in tell() metode.

```

with open("test.txt", "r") as f:
    print(f.tell()) # get current position of file cursor in
characters
    f.read(4) # read 4 bytes
    print(f.tell())

```

```

0
0

```

```

with open("test.txt", "r") as f:
    print(f.tell()) # get position in bytes

    reading = f.read(6) # read 6 bytes
    print(reading)
    print(f.tell()) # get new position in bytes

    f.seek(0) # move cursor to position 0
    print(f.tell())

    reading = f.read(6)
    print(reading)

```

```

0

0
0

```

Datoteko lahko hitro in učinkovito preberemo vrstico po vrstico, z uporabo for loop.

```

with open("test.txt", "r") as f:
    for line in f:

```

```
'\n'.    print(line) # The lines in file itself has a newline character
```

Alternativno lahko uporabljamo `readline()` metodo za branje individualnih vrstic.

Metoda prebere podatke iz datoteke do `newline (\n)`.

```
with open("test.txt", "r") as f:
    print(f.readline())
    print(f.readline())
    print(f.readline())
```

`readlines()` nam vrne listo preostalih linij v datoteki.

(če prov vidm `readlines()` prebere vrstice in postavi cursor na konc)

```
with open("test.txt", "r") as f:
    list_of_lines = f.readlines()
    print(list_of_lines)
    print(list_of_lines[1])
```

```
[]
```

```
-----
-----
```

```
IndexError                                Traceback (most recent call
last)
c:\Users\nkast\ICTA\python-osnovni-public\Termin06\
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb Cell 46 line
4
```

```
    <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X63sZmlsZQ
%3D%3D?line=1'>2</a> list_of_lines = f.readlines()
    <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X63sZmlsZQ
%3D%3D?line=2'>3</a> print(list_of_lines)
----> <a href='vscode-notebook-cell:/c%3A/Users/nkast/ICTA/python-
osnovni-public/Termin06/
P5_Prenovljeno_Datoteke_Importing_Errors_Matplotlib.ipynb#X63sZmlsZQ
%3D%3D?line=3'>4</a> print(list_of_lines[1])
```

```
IndexError: list index out of range
```

```
with open("test.txt") as f:
    for line in f.readlines():
        print(line)
```

INPUT:  
funkcija(3)

OUTPUT:  
line 7  
line 8  
line 9

```
def funkcija(n):
    with open("nalog2.txt", "r") as f:
        data = f.readlines()
        for line in data[-n:]:
            print(line, end="")
```

funkcija(3)

line 7  
line 8  
line 9

INPUT:  
dictionary()

OUTPUT:  
Vnesi besedo: meow  
homeown  
homeowner  
homeowners  
meow  
meowed  
meowing  
meows

```
def dictionary():
    beseda = input("Vnesi besedo: ")

    with open("words_alpha.txt", "r") as f:
        for line in f.readlines():
            if beseda in line:
                print(line, end="")
```

dictionary()

---

## Pisanje datotek

Za pisanje v datoteko jo odpremo v načinu za pisanje:

- **w** (ta način bo prepisal vse podatke že shranjene v datoteki)
- **a** (s tem načinom bomo dodajali podatke na konec datoteke)
- **x** (s tem ustvarimo datoteko in lahko začnemo v njo pisati)

Writing a string or sequence of bytes (for binary files) is done using `write()` method. This method returns the number of characters written to the file.

```
with open("test.txt", 'w') as f:
    f.write("my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")

# This program will create a new file named 'test.txt' if it does not exist. If it does exist, it is overwritten.
# We must include the newline characters ourselves to distinguish different lines.

with open("test.txt", 'a') as f:
    f.write("We are adding another line.")
    x = f.write("And another one")
    #print(x) write() returns number of bytes we wrote

# this program will open a file and append what we want to the end

with open("test2.txt", "x") as f:
    f.write("New .txt")

# this program will create a new file 'test2.txt' if it doesn't exist and write into it.
# if the file exists it will throw an error
```

---

## Importing

**Importing** je način, kako lahko kodo iz ene datoteke/modula/package uporabimo v drugi datoteki/modulu.

- **module** je datoteka, ki ima končnico `.py`
- **package** je direktorij, ki vsebuje vsaj en modul

Da importiramo modul uporabimo besedo `import`.

```
import moj_modul
```



Python sedaj prvo preveri ali se *moj\_modul* nahaja v **sys.modules** - to je dictionary, ki hrani imena vseh importiranih modulov.

Če ne najde imena, bo nadaljeval iskanje v **built-in** modulih. To so moduli, ki pridejo skupaj z inštalacijo Pythona. Najdemo jih lahko v Python Standardni Knjižnici - <https://docs.python.org/3/library/> .

Če ponovno ne najde našega modula, Python nadaljuje iskanje v **sys.path** - to je list direktorijev med katerimi je tudi naša mapa.

Če Python ne najde imena vrže **ModuleNotFoundError**. V primeru, da najde ime, lahko modul sedaj uporabljamo v naši datoteki.

---

Za začetek bomo importiral **math** built-in modul, ki nam omogoča naprednejše matematične operacije, kot je uporaba korenjenja.

math documentation - <https://docs.python.org/3/library/math.html>

Da pogledamo katere spremenljivke / funkcije / objekti / itd. so dostopni v naši kodi lahko uporabimo **dir()** funkcijo.

dir documentation - <https://docs.python.org/3/library/functions.html#dir>

```
import math

moja_spremenljivka = 5
print(dir())

print(moja_spremenljivka)
print(math)
```

S pomočjo **dir(...)** lahko tudi preverimo katere spremenljivke, funkcije, itd. se nahajajo v importiranih modulih.

```
import math

moja_spremenljivka = 5
print(dir(math))
```

Funkcijo, spremenljivko, atribut v math modulu uporabimo na sledeč način:

```
import math

print(math.sqrt(36))

# Rešitev
import math
```

```
math.log(144, 12)
2.0
```

## Importing our own module

Ustvarimo novo datoteko **moj\_modul.py** zraven naše datoteke s kodo.

moj\_modul.py

```
class Pes():
    def __init__(self, ime):
        self.ime = ime

def sestevalnik(a, b):
    return a+b

moja_spremenljivka = 100
```

skripta.py

```
import moj_modul

print(dir())
print(dir(moj_modul))

fido = moj_modul.Pes("fido")
print(fido.ime)

print(moj_modul.sestevalnik(5, 6))

print(moj_modul.moja_spremenljivka)
```

## Načini importiranja

Importiramo lahko celotno kodo ali pa samo specifične funkcije, spremenljivke, objekte, itd.

Celotno kodo importiramo na sledeči način:

```
import moj_modul

import moj_modul

print(dir())
```

```
fido = moj_modul.Pes("fido")
print(fido.ime)

print(moj_modul.sestevalnik(5, 6))
print(moj_modul.moja_spremenljivka)
```

Specifične zadeve importiramo na sledeč način:

```
from moj_modul import moja_spremenljivka
from moj_modul import moja_spremenljivka

print(dir())
print(moja_spremenljivka)

from moj_modul import sestevalnik

print(dir())
print(sestevalnik(5,6))

from moj_modul import Pes

print(dir())
fido = Pes("fido")
print(fido.ime)
```

---

Importirane zadeve se lahko shrani tudi pod drugim imenom

```
import moj_modul as mm
import moj_modul as mm

print(dir())

fido = mm.Pes("fido")
print(fido.ime)

print(mm.sestevalnik(5, 6))
print(mm.moja_spremenljivka)

from moj_modul import sestevalnik as sum_

print(dir())
print(sum_(5,6))
```

Za premikanje med direktoriji med importiranjem se uporablja ". " .

```

from package1.module1 import function1

# test.py
import naloga1

r1 = naloga1.pretvornik(180, mode="deg2rad")
if float(str(r1)[:4]) == 3.14:
    print("Rešitev pravilna.")
else:
    print("Nekaj je narobe.")

r2 = naloga1.pretvornik(360, mode="deg2rad")
if float(str(r2)[:4]) == 6.28:
    print("Rešitev pravilna.")
else:
    print("Nekaj je narobe.")

r3 = naloga1.pretvornik(1.5707963267948966, mode="rad2deg")
if r3 == 90:
    print("Rešitev pravilna.")
else:
    print("Nekaj je narobe.")

r3 = naloga1.pretvornik(4.71238898038469, mode="rad2deg")
if r3 == 270:
    print("Rešitev pravilna.")
else:
    print("Nekaj je narobe.")

# Rešitev
import math

def pretvornik(x, mode="deg2rad"):
    if mode == "deg2rad":
        return math.radians(x)
    elif mode == "rad2deg":
        return math.degrees(x)

```

---

Importiramo lahko tudi vse naenkrat z uporabo "\*" vendar se to odsvetuje, saj ne vem kaj vse smo importirali in lahko na tak način ponesreči kaj spremenimo.

```

#Ime datoteke naloga1.py
from math import *

print(dir())
print(pi)

```

```
pi = 3
print(pi)
```

---

```
13. Mar 2020
13. Nov 2020
13. Aug 2021
13. May 2022
13. Jan 2023
13. Oct 2023
13. Sep 2024
13. Dec 2024
13. Jun 2025
13. Feb 2026
13. Mar 2026
13. Nov 2026
13. Aug 2027
13. Oct 2028
13. Apr 2029
13. Jul 2029

from datetime import date

def funkcija():
    with open("naloga_petek13.txt", "w") as f:
        for year in range(2020, 2030):
            for month in range(1, 13):
                datum = date(year, month, 13)
                #print(datum)
                if datum.weekday() == 4: # 0=Mon, 1=Tue, ..., 4=Fri
                    f.write(f'{datum.strftime("%d. %b %Y")}\n')

funkcija()
```

---

## JSON

<https://realpython.com/python-json/>

JSON - JavaScript Object Notation, je način zapisa informacij v organizirano in preprosto strukturo, ki je lahko berljiva tako za ljudi kot tudi za računalnike.

Če boste ustvarjali program, ki zajemajo podatke iz interneta, boste JSON velikokrat videli.

```
{
    "firstName": "Jane",
```

```

{
  "lastName": "Doe",
  "hobbies": ["running", "sky diving", "singing"],
  "age": 35,
  "children": [
    {
      "firstName": "Alice",
      "age": 6
    },
    {
      "firstName": "Bob",
      "age": 8
    }
  ]
}

```

Za manipuliranje z JSON podatki v Pythonu uporabljamo `import json` modul.

## Python object translated into JSON objects

Primer shranjevanja JSON podatkov.

```

import json

data = {
    "firstName": "Jane",
    "lastName": "Doe",
    "hobbies": ("running", "sky diving", "singing"),
    "age": 35,
    "children": [
        {
            "firstName": "Alice",
            "age": 6
        },
        {
            "firstName": "Bob",
            "age": 8
        }
    ]
}

```

Da naše podatke spremenimo v JSON zapis uporabimo metodo `json.dumps(data)`.

Nazaj dobimo string katerega bi lahko zapisali v JSON datoteko.

```

json_data = json.dumps(data)
json_data

```

```
'{"firstName": "Jane", "lastName": "Doe", "hobbies": ["running", "sky diving", "singing"], "age": 35, "children": [{"firstName": "Alice", "age": 6}, {"firstName": "Bob", "age": 8}]}'
```

Da JSON string pretvorimo nazaj v python datatipe uporabimo funkcijo `json.loads(json_string)`.

```
py_data = json.loads(json_data)

print(py_data)
py_data["firstName"]

{'firstName': 'Jane', 'lastName': 'Doe', 'hobbies': ['running', 'sky diving', 'singing'], 'age': 35, 'children': [{'firstName': 'Alice', 'age': 6}, {'firstName': 'Bob', 'age': 8}]}

'Jane'
```

---

Če želimo podatke direktno shraniti v `.json` datoteko imamo za to metodo `json.dump()`.

```
with open("data_file.json", "w") as write_file:
    json.dump(data, write_file)
# Note that dump() takes two positional arguments:
#(1) the data object to be serialized,
#and (2) the file-like object to which the bytes will be written.
```

Da podatke preberemo nazaj iz datoteke imamo metodo `json.load()`.

Potrebno se je zavedati, da konverzija datatipov ni exactna (ni točna). To pomeni, če smo v začetnih podatkih imeli neke tuple in ga shranili v JSON in ta JSON nato prebrali nazaj v python, tuple postane list.

### Python-JSON conversion table

```
with open("data_file.json", "r") as read_file:
    data = json.load(read_file) # use loads() if the JSON data is in
"python string" type
    print(data)
    print(type(data))

{'firstName': 'Jane', 'lastName': 'Doe', 'hobbies': ['running', 'sky diving', 'singing'], 'age': 35, 'children': [{'firstName': 'Alice', 'age': 6}, {'firstName': 'Bob', 'age': 8}]}
<class 'dict'>
```

OUTPUT:

Oseba, ki zasluži največ je martha. Zasluži 10300€.

```

import json

with open("podatki.json") as f:
    data = json.load(f) # use loads() if the JSON data is in "python
                        # string" type
                        #print(data)

max_pay = 0
name = ""
for employee in data["company"]["employees"]:
    print(employee)
    if employee["payble"]["salary"] + employee["payble"]["bonus"] >
max_pay:
    name = employee["name"]
    max_pay = employee["payble"]["salary"] + employee["payble"]
["bonus"]

print(f"Oseba, ki zasluži največ je {name}. Zasluži {max_pay}€.")

{'name': 'emma', 'payble': {'salary': 7000, 'bonus': 800}}
{'name': 'derek', 'payble': {'salary': 4000, 'bonus': 1000}}
{'name': 'alex', 'payble': {'salary': 7500, 'bonus': 500}}
{'name': 'susan', 'payble': {'salary': 6300, 'bonus': 350}}
{'name': 'martha', 'payble': {'salary': 9100, 'bonus': 1200}}
{'name': 'clark', 'payble': {'salary': 7700, 'bonus': 270}}
{'name': 'luise', 'payble': {'salary': 8200, 'bonus': 900}}
Oseba, ki zasluži največ je martha. Zasluži 10300€.

```

## Titanic Analysis

Za našo nalogo si bomo sedaj pogledali podatke o potnikih ladje Titanic - [titanic.csv](#).

Pogledali si bomo koliko preživelih potnikov je bilo moških in koliko žensk

Podatki so shranjeni v sledeče:

- **PassengerId** - ID vsakega potnika. Da lahko ločimo potnike med seboj
- **Survived** - 0, če oseba ni preživela. 1, če je oseba preživela.
- **Pclass** - v katerem razredu je bila oseba. 1 - prvi razred, 2 - drugi razred, 3 - tretji razred
- **Name** - ime osebe
- **Last name** - priimek osebe
- **Sex** - spol osebe
- **Age** - starost osebe v letih
- **SibSp** - število sorojencev osebe oziroma partnerjev na ladji
- **Parch** - število staršev oziroma otrok osebe na ladji



- **Ticket** - številka karte osebe
- **Fare** - cena karte
- **Cabin** - številka sobe
- **Embarked** - kje se je oseba vkrcala, C = Cherbourg, Q = Queenstown, S = Southampton

Prvih 5 vrstic v listu `data`:

```
[ '1', '0', '3', '"Braund', ' Mr. Owen Harris"', 'male', '22', '1',
'0', 'A/5 21171', '7.25', '', 'S' ]
[ '2', '1', '1', '"Cumings', ' Mrs. John Bradley (Florence Briggs
Thayer)', 'female', '38', '1', '0', 'PC 17599', '71.2833', 'C85',
'C' ]
[ '3', '1', '3', '"Heikkinen', ' Miss. Laina"', 'female', '26', '0',
'0', 'STON/O2. 3101282', '7.925', '', 'S' ]
[ '4', '1', '1', '"Futrelle', ' Mrs. Jacques Heath (Lily May Peel)',
'female', '35', '1', '0', '113803', '53.1', 'C123', 'S' ]
[ '5', '0', '3', '"Allen', ' Mr. William Henry"', 'male', '35', '0',
'0', '373450', '8.05', '', 'S' ]
```

```
data = []
```

```
with open("./titanic.csv", "r") as f:
    for line in f.readlines()[1:]: # preskočimo prvo vrstico, ker so
        to imena stolpcev
        line = line.strip() # odstranimo nepotrebne \n in presledke
        line_splitted = line.split(",") # razbijemo vrstico na
        posamezne podatke
        data.append(line_splitted)
```

```
for line in data[:5]:
    print(line)
```

```
[ '1', '0', '3', '"Braund', ' Mr. Owen Harris"', 'male', '22', '1',
'0', 'A/5 21171', '7.25', '', 'S' ]
[ '2', '1', '1', '"Cumings', ' Mrs. John Bradley (Florence Briggs
Thayer)', 'female', '38', '1', '0', 'PC 17599', '71.2833', 'C85',
'C' ]
[ '3', '1', '3', '"Heikkinen', ' Miss. Laina"', 'female', '26', '0',
'0', 'STON/O2. 3101282', '7.925', '', 'S' ]
[ '4', '1', '1', '"Futrelle', ' Mrs. Jacques Heath (Lily May Peel)',
'female', '35', '1', '0', '113803', '53.1', 'C123', 'S' ]
[ '5', '0', '3', '"Allen', ' Mr. William Henry"', 'male', '35', '0',
'0', '373450', '8.05', '', 'S' ]
```

```
{ 'male': { 'survived': 109,
            'died': 468,
            'survived_%': 0.18890814558058924,
            'died_%': 0.8110918544194108 },
  'female': { 'survived': 233,
```

```

        'died': 81,
        'survived_%': 0.7420382165605095,
        'died_%': 0.25796178343949044}}

survived_dist = {"male": {"survived": 0, "died": 0}, "female":
{"survived": 0, "died": 0}}

for line in data:
    if int(line[1]): # person survived
        survived_dist[line[5]]["survived"] += 1
    else:
        survived_dist[line[5]]["died"] += 1

survived_dist

{'male': {'survived': 109, 'died': 468},
 'female': {'survived': 233, 'died': 81}}

for s in survived_dist:
    total = survived_dist[s]["survived"] + survived_dist[s]["died"]
    survived_dist[s]["survived_%"] = survived_dist[s]["survived"] /
total
    survived_dist[s]["died_%"] = survived_dist[s]["died"] / total

survived_dist

{'male': {'survived': 109,
 'died': 468,
 'survived_%': 0.18890814558058924,
 'died_%': 0.8110918544194108},
 'female': {'survived': 233,
 'died': 81,
 'survived_%': 0.7420382165605095,
 'died_%': 0.25796178343949044}}

```

---