

10_files and exceptions

- reading from a file
 - reading a n entire file
 - line by line
 - file path
 - relative
 - absolute
 - store each line as a list
 - work with files' content
- writing to a file
 - to an empty file
 - append to a file
- Exception
 - ZeroDivisionError
 - FileNotFoundError
 - fails silently
- storing data
 - store
 - load
 - combine it with input
 - name remembering machine

reading from a file

reading a n entire file

in the first step, i created a file named "pie.txt" containing the first 30 digits of pie.

```
with open('pi.txt') as pi:  
    pi_in30_digits=pi.read()  
print(pi_in30_digits)  
# in order to view the data of the file, we open the file and then use read() to  
read the data.
```

line by line

```
file_path = "C:/Users/Marco/Downloads/transcript.txt"
```

```
with open(file_path) as file_object:
    for line in file_object:
        print(line)
```

```
file_path="C:/Users/Marco/Downloads/transcript.txt"
file=open(file_path)
    for line in file
        print(line)
```

file path

relative

if you don't provide the full file path of your targeted file, python will only look at its own directory. to solve this issue, one may can provide your program with a file path that is relative to the current directory. For example, if the txt file we are looking for is stored in a folder, text_files, which is stored in the directory python_work:

```
with open('text_files/pi.txt') as fileobject
```

By specifying the relative file path, python is now able to go into the text_files folder and find our targeted file.

absolute

With an absolute file path, we can locate any file in any position.

```
file_path="C:/Users/Marco/Downloads/transcript.txt"
with open(file_path)as transcript
    transcript_content=transcript.read()
print(transcript_content)
```

store each line as a list

```
file_path="C:/Users/Marco/Downloads/transcript.txt"
with open(file_path) as transcript:
    lines=transcript.readlines()
for line in lines:
    print(line)
```

work with files' content

```

file_path="C:/python_work/pi.txt"
with open(file_path) as pi:
    lines=pi.readlines()
real_pi=''
for line in lines:
    real_pi+=line.strip()
print(real_pi)
print(len(real_pi))

```

writing to a file

to an empty file

```

newfile="i_feel_marvelous_today"
with open(newfile,'w') as marvelous_edit:
    marvelous_edit.write('what a marvelous day')
with open(newfile) as marvelous_edit:
    lines=marvelous_edit.read()
print(lines)
# if the file you are written does exist, it will erase all the content it has and
rewrite as you instructed, if the file does not exist, a new file will be created
automatically.
#'w' means writing mode.'r':reading mode,'a':append mode,'r+':read and write mode.

```

append to a file

```

# there are circumstances when we want to add lines to an existing files instead of
rewriting it.
newfile="i_feel_marvelous_today"
with open(newfile,'a') as marvelous_add_new:
    marvelous_add_new.write('\ntomorrow is gonna be marvelous too!')
with open(newfile) as marvelous_add_new:
    lines=marvelous_add_new.read()
print(lines)
# what a marvelous day
# tomorrow is gonna be marvelous too!

```

Exception

python may stop running when it encounters something unexpected. However, it can be directed to a series of other instruction if we create an exception block.

ZeroDivisionError

python will report an error when we try to divide something by zero. when this happens, we can use try-except block to handle.

```
print('type your 2 numbers and i will divide them')
while True:
    first_number=input("type your first number")
    if first_number==q:
        break
    second_number=input("type your second number here")
    if second_number==q:
        break
    try:
        answer=int(first_number)/int(second_number)
    except ZeroDivisionError:
        print("bro, you can't divide zero")
    else:
        print(answer)
```

FileNotFoundError

```
file_name=input('type your file name here')
try:
    with open(file_name) as f:
        content=f.read()
except FileNotFoundError:
    print(f'sorry,the file {file_name} does not exist')
```

fails silently

```
#to pretend nothing happened
file_name=input('type your file name here')
try:
    with open(file_name) as f:
        content=f.read()
except FileNotFoundError:
    pass
```

storing data

json module allows us to store the data user input and load it when it is required.

store

```
import json
numbers=[1,2,3,5,6,8,9]
user_file='numberlist.json'
with open(user_file,'w') as u:
    json.dump(numbers,u)
```

load

```
import json
with open(user_file) as u:
    numbers=json.load(u)
    print(numbers)
```

combine it with input

```
import json
username=input('type your username below')
file_name='name_collection'
with open(file_name,'w') as f:
    json.dump(username,f)
print(f'your username {username} has been saved!')
```

name remembering machine

```
import json
file_name='users_firstname.json'
try:
    with open(file_name) as f:
        username=json.load(f)
except FileNotFoundError:
    username=input('hi waht is your name?')
    with open(file_name,'w') as f:
        json.dump(username,f)
        print(f'ok {username},your name is remembered')
else:
    print(f'hi {username}')
```