Welcome to Lecture number 12, pythonistas. This is Akshansh (+91, 8384891269, akshanshofficial@gmail.com (mailto:akshanshofficial@gmail.com)) and we are here to discuss about the file handling in python.

Let's check our directory

In [1]:

```
pwd
```

Out[1]:

```
'/home/akshansh'
```

My current working directory is /home/akshansh

let's set our current directory to Dekstop

In [2]:

```
cd Desktop/
```

```
/home/akshansh/Desktop
```

I've set my current directory to Desktop, you can check it too by using pwd.

In [4]:

```
pwd
```

Out[4]:

```
'/home/akshansh/Desktop'
```

as you can now see that my current directory is Desktop now. You guys must be caught by a tought that why I set Desktop my current directory (it's my choice :-p). Well, I have chosen this because it is easy to locate files in desktop. I suggest y'all to do the same so that we can make codes together.

Let's read a file from directory.

Before reading a file from directory, we must have a file there. Since you are working with me, I want you all to creat a same file like i did. In this case you have to create a .txt file and we will be working on that file.

so how you create a .txt file?

you have to search it how to create a .txt file on your PC. In linux you can simply go to text editor and then add the .txt after name fo the file.

```
Suppose you have created a .txt file. Kindly name it pi_digits.txt and save
value of pi in it.
```

```
3.1415926535
8979323846
2643383279
```

let's read this file. Make sure that the file pi_digits.txt is in your Desktop directory

In [5]:

```
with open("pi_digits.txt") as file_object:
    contents=file_object.read()
print(contents)
```

```
3.1415926535
  8979323846
  2643383279
```

OBSERVATIONS-

open() method is being used to open file in directory. We write >>>with open("name of file here") as file_object:

we will be using with open("pi_digits.txt") as file_object

: will be used in the end

in next line I wrote, contents=file_object.read(). read() method is being used to read the contents(whatever in that file) in the file.

print(contents) will print all the contents in the file pi_digits.txt

In [6]:

```
#let's see the contents
print(contents)
```

```
3.1415926535
  8979323846
  2643383279
```

you can see that extra steps in the end here, we can delete extra steps in rstrip() method.

In [8]:

```python
with open("pi_digits.txt") as file_object:
    contents=file_object.read()
print(contents.rstrip())
```

```
3.1415926535
   8979323846
   2643383279
```

You can also access the file with absolute path.

In my case, file pi_digits.txt is in Desktop folder which is in (user) akshansh and which is in home directory

In [9]:

```python
file_path='/home/akshansh/Desktop/pi_digits.txt'
with open(file_path) as file_object:
    contents=file_object.read()
print(contents)
```

```
3.1415926535
   8979323846
   2643383279
```

OBSERVATIONS

I created file_path, location of the folder where my desired file stays. And then, i stored the location/path in a variable name file_path.

while you pass argument in open() method, instead of passing file name we pass file_path. And everything remains same.

when you're reading a file, you'll often want to examine each line of the file. You might be looking for the certain information in the file, you might want to modify the text in the file somway.

In [10]:

```python
filename='pi_digits.txt'
with open(filename) as file_object:
    for line in file_object:
        print(line)
```

```
3.1415926535

   8979323846

   2643383279
```

these blank lines appear becuase an invisible new line character is at the end of each line in text file. You can use rstrip() on each line in the print() canll elininates these extra blank lines.

In [11]:

```
filename='pi_digits.txt'
with open(filename) as file_object:
    for line in file_object:
        print(line.rstrip())
```

```
3.1415926535
   8979323846
   2643383279
```

You can see that, extra line has gone now.

making a list of lines from a file

when you use with open(), the file_object returns, is only available inside the with block that contains it. If you want to retain access to a file's contents outside the with block, you can store file's lines in a list indide the block and then work with that list. You can process parts of the gile immediately and postpone some processing for later in the program.

In [12]:

```
filename="pi_digits.txt"
with open(filename) as file_object:
    lines=file_object.readlines()

for line in lines:
    print(line.rstrip())
```

```
3.1415926535
   8979323846
   2643383279
```

readlines() method takes each line of the file and then saves it into the variable named called lines(you decide that)

working with a file's content

after you've read that file into the memory, you can do whatever you want with that data, so let's briefly explore the digits of pi.

First we will attempt to build a single string containing the the digits in the file with no whitespace in it:

In [14]:

```python
filename="pi_digits.txt"
with open(filename) as file_object:
    lines=file_object.readlines()

pi_string= ''
for line in lines:
    pi_string+=line.rstrip()

print(pi_string)

```

```
3.1415926535    8979323846    2643383279
```

In [15]:

```python
print(len(pi_string))
```

```
39
```

we created a variable, pi_string, to hold digits of the pi. We then create a loop that adds each line of digits to pi_strinf and removes the newline character from each line. Then we print thhis string and also show how long string is

when python reads from a text file, it interprets all the text in the file as a string. If you read in a number and want to work with that value in a numerical context, you'll have to convert it to an integer using int() finction or convert it to a float using the float() function

writing to a file

one of the simplest way to save data is to write it to a file. When you write text to a file, the output will still be available after you close the terminal containing your program's output. You can examine output after a program fisnishes running and you can share the outpuy fikes with others as well.

In [16]:

```python
filename="programming.txt"
with open(filename,'w') as file_object:
    file_object.write("I love programming.")
```

check you Desktop folder(current directory) there must be a file named programming.txt

try to open it and you'll find that in this file I love programming.

there are three modes in python to play with files - read mode ('r'), write mode ('w') and append mofe('a') and one another specific method/mode is also there that allows you to read and write ('r+')

python can only write strings to a text file. If you want to store numerical data in a text file, you'll have to convert the data to string by str() method

writing multiple lines

In [17]:

```python
filename="programming.txt"
with open(filename,'w') as file_object:
    file_object.write("I love programming.")
    file_object.write("I love creating new games.")
```

for seperate lines use \n to enter a new line

check the that file named programming.txt again, a new line has been added there.

appending to a existed file

if you want to add content to a file instead of writing a new one. You can open file in append mode. When you open a file in append mode, python doesn't erase the contents of the file. It simply adds at the end of the file.

In [18]:

```python
filename="programming.txt"
with open(filename,'a') as file_object:
    file_object.write("\nthis line has been added by using append method.\n")
    file_object.write("this is 2nd line by append method.\n")
```

Check that file again, file has been updated there too.

This is how you can read a file or write and add some more texts in existing file. Hope you guys found it helpful. This was Akshansh, and keep pythoning. Have a great day.

In [ ]:

```python

```