Hi guys, this is Akshansh (+91 8384891269, [akshanshofficial@gmail.com (mailto:akshanshofficial@gmail.com)](mailto:akshanshofficial@gmail.com)) and we are here to discuss lecture 13 about the tracebacks and errors/exceptions.

exceptions

python uses special objects called exceptins to manage errors that arises during a protram's execution. Whenever an error occurs that makes python unsure what to do next, it creates an exception object. If you write code that handles the exception, the program will continue running. If you are not able to handle the errors, your program will get into error and this is how you will be called bad programmer.

exception are handled with try except blocks.A try except block asks python to do something, but it also tells puthon what to do if an exception is raised. When you use try except blocks, your programs will continue running even if thing start to go wrong.

handling the ZeroDivisionError Exception

In [3]:

```
print(5/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-3-fad870a50e27> in <module>
----> 1 print(5/0)

ZeroDivisionError: division by zero
```

you see that, you can't divide 5 by 0, pyhton gives you an error ZeroDivisionError. ZeroDivisionError is an exception object in try except block in python. It stops program and rest of the code after that will not be executed.

using try-except blocks

when you think that a problem can occur, you can write a try-exceot block to handle the exception that might be raised. You tell python to try running some code and you tell it what to do next.

In [4]:

```
try:
    print(5/0)
except ZeroDivisionError:
    print("You can't divide by 0, python wants to know you school location")
```

```
You can't divide by 0, python wants to know you school location
```

using exception to prevent crashes

handling errors correctly is especially important whnen the program has more work to do after the error occurs. This happens often in the programs that prompts user for input. If the program responds to invalid input appropriatelt, it can prompt for more valid instead of crashing. Let's create a calculator that does only division

In [5]:

```
print("give me two numbers and I'll divide them.")
print("enter q to quit")

while True:
    firstNumber=input("\nFirst number: ")
    if firstNumber=='q':
        break
    secondNumber=input("Second number: ")
    if secondNumber=='q':
        break
    answer=int(firstNumber)/int(secondNumber)
    print(answer)
```

```
give me two numbers and I'll divide them.
enter q to quit

First number: 10
Second number: 2
5.0

First number: 15
Second number: 0

----------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
 last)
<ipython-input-5-a089952099a7> in <module>
      9     if secondNumber=='q':
     10         break
---> 11     answer=int(firstNumber)/int(secondNumber)
     12     print(answer)

ZeroDivisionError: division by zero
```

it's bad that the program crashed but it is also not a good idea to let user see tracebacks. Notechnical user will ve confused by them,and in a malicious setting. attackers will learn more than you want them to know from a tracebacks. For example, they'll know the file name of your programme and the code causing trouble.

the else block

we can make this program more error resistant by wrapping the line that might produce errors in a try except vlock. THe error occurs on the line that performs division, so that's where we'll put the try except block.

In [6]:

```python
print("give me two numbers and I'll divide them.")
print("enter q to quit")

while True:
    firstNumber=input("\nFirst number: ")
    if firstNumber=='q':
        break
    secondNumber=input("Second number: ")
    if secondNumber=='q':
        break
    try:
        answer=int(firstNumber)/int(secondNumber)
    except ZeroDivisionError:
        print("No one divides by 0.")
    except:
        print("have you ever seen someone using letter to divide?")
    else:
        print(answer)
```

```
give me two numbers and I'll divide them.
enter q to quit

First number: 10
Second number: 2
5.0

First number: 10
Second number: 0
No one divides by 0.

First number: 23
Second number: r
have you ever seen someone using letter to divide?

First number: q
```

Handling file not found errors

one common issue we find when working with the file is missing file in our directory. How to deal with that? It may be possible that either file is not in your device or not in the current directory. Let's try to read a file that doesn't exist.

In [7]:

```python
filename="DNEfile"
with open(filename) as file_object:
    content=filename.read()
    print(content)
```

```
-----------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
 last)
<ipython-input-7-c0fb5164290a> in <module>
      1 filename="DNEfile"
----> 2 with open(filename) as file_object:
      3     content=filename.read()
      4     print(content)

FileNotFoundError: [Errno 2] No such file or directory: 'DNEfile'
```

you can see that error/exception raised in this case is FileNotFoundError. Let's deal with it. Suppose there is a file that does not exist in you system. I named that file as DNEfile.

In [8]:

```python
filename="DNEfile"
try:
    with open(filename) as file_object:
        content=filename.read()

except FileNotFoundError:
    print(f"{filename} is not in your PC or current directory. Make sure it is

else:
    print(content)
```

```
DNEfile is not in your PC or current directory. Make sure it is there.
```

Analyzing text

you can analyze text files containing entire books. Many classic works of literature are available as simple text files because they are in public domain. You can use any of them and what interesting thisng you can do that you can knwo how many words are there in that book.

In [9]:

```python
title="Alice in Wonderland"
title.split()
```

Out[9]:

```
['Alice', 'in', 'Wonderland']
```

split() separate items on the basis of space and store them in a list.

now you can count the how many words a list has. or a text file has

In [10]:

```
len(title)
```

Out[10]:

```
19
```

working with multiple files

review lecture notes number 12, for more info about the directory path

In [11]:

```
cd Desktop/
```

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

In [12]:

```
pwd
```

Out[12]:

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

In [13]:

```python
def count_words(file):
    """Count the approximate number of words in a file."""
    try:
        with open(file) as file_object:
            contents = file_object.read()
    except FileNotFoundError:
        print(f"Sorry, the file {file} does not exist.")
    else:
        words = contents.split()
        num_words = len(words)
        print(f"The file {file} has about {num_words} words.")


filenames=["programming.txt","pi_digits.txt","DNEfile"]

for filename in filenames:
    print(count_words(filename))

```

```
The file programming.txt has about 23 words.
None
The file pi_digits.txt has about 3 words.
None
Sorry, the file DNEfile does not exist.
None
```

Thanks for your time, my time is up now. Have a blessed life. Stay safe and away from Covid-19. Thanks you guys.

In [ ]:

```

```