**Generators:**

In our previous class, we have seen iterators. If we want to print values one by one for example from 1 to 100 then we will use iterators which we have seen in our previous class.

Complications with iterator are we need to create it, where we have to create 2 methods iter() and next().

Generator will give iterators. Let’s look at how to create generators and start by creating a ***function*** not a class/method. Function to return the top ten values. We will use special keyword ***yield*** which will make the function a generator.

Graphical user interface, text, application

Description automatically generated

By running the above code instead of printing the value 5, we got the object of generator. As we have already discussed generator will give iterator so the function topten() is NOT a normal function it is a generator because we are using yield. **If you change the yield word to return it will print 5 as a normal function would do.**

* As per last class if you want to fetch something from iterator we need to use the \_\_next\_\_ function, run the below code and we will get 5

Graphical user interface, text, application, email

Description automatically generated

* Since yield is a generator which will give us iterator we can write it multiple **yield statements.** If we run this code we will only get the first value 1 as we have mentioned \_\_next\_\_ only once as shown below. Run the code and observe the output.

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* We can use a loop as well as shown below and it will pick for the remaining values 3,4,5

Graphical user interface, text, application

Description automatically generated

* We can take another example here, like print top ten perfect square

Graphical user interface, text, application

Description automatically generated

Q: why someone will use a generator: let’s say you are fetching 1000 records from a database, may be you want to print all or process something from those records. So when you fetch these 1000 records they will be loaded in your memory, we don’t want that and may be you want to work with one value/record at a time. In such scenario, we will use generator instead of fetching entire list you can fetch one value at a time.

**Intro to exception handling:**

Before exception handling we need to know about errors. There are 3 types of errors which are shown in the below screenshot.

**A picture containing text, person, person, indoor

Description automatically generated**

1. Syntax errors: which are like missing : for if statement or any spelling mistake. These type of errors were found during the compiling hence they are called compile time errors.

Timeline

Description automatically generated

1. Logical errors: syntax is good and we have a mistake with logic, example we want to add two numbers and mentioned –(minus) in the print statement which will give the inaccurate/wrong output.

Timeline

Description automatically generated with medium confidence

1. Runtime error: you code gets compiled and it has proper syntax and it is not giving logical error as well. Ex: 2/2 = 1, 4/2 = 2. What if the user is giving 6/0 . code gets compiled, no logical error and it is not working for specific error which is during the runtime.

Which is the most easiest error to find out?

Logical errors? Are they related to bugs?

What about runtime error? – user side issue.

As a developer, we should think of all the possible errors that can occur and take proactive measures.

Software should not stop working because of the errors as there might be a loss for business- think of a bank software. Have a casual discussion about the administrators, software developers, network, databases.

Timeline

Description automatically generated

Let’s take a look with an example about the

Graphical user interface, application

Description automatically generated with medium confidence

* Below screenshot give zero division error.

Graphical user interface, text, application

Description automatically generated

Problems from above code:

1) User will not understand what this error means

2) we are not getting bye from the above code.

* Let’s fix this by using special words ***try*** and ***except Exception*** as shown below and observe the output.

Graphical user interface, application

Description automatically generated

So what we have done above is we are trying to execute the print and if it works we are print it, else we are going to print the except block.

* We will try dividing normal numbers and check the output, ex as shown below.

Graphical user interface, text, application

Description automatically generated

* Now what if we want to print the error in case if we get one.

Graphical user interface, text, application, email

Description automatically generated

* Now let’s look at something related to file open and close, database connection open and close before we exit the program. It is a good practice. Here let’s say we are opening a resource/file in the try block and closing it in the same the below output occurs.

Graphical user interface, text, application

Description automatically generated

* What if we get the exception as we divide something by zero, the resource is opened and not closed as shown below. It is during the print(a/b) we are getting the exception and the code is jumping out of the try and calling the exception hence it is not able to close the resource which is the print statement

Graphical user interface, application

Description automatically generated

* How to fix it? – we will close the resource after the exception as shown below

Graphical user interface, text, application

Description automatically generated

* Now if we change the value of second variable from zero to two, now in this case if we observe the resource is not closed as per the below screenshot. It is not executing the exception block as it don’t have the error. To fix this we can use **finally**

Graphical user interface, text, application

Description automatically generated

* ***Finally*** block will get executed if we get error as well as if we don’t get error. Remember except block will get executed ONLY when there is an error.

***Graphical user interface, text, application

Description automatically generated***

* Now let’s try adding a input function to our code and see how it works.

***Graphical user interface, text, application

Description automatically generated***

* Let’s input the character/alphabet rather than an int and observe the output, it gives zero division error and “invalid lateral……p” which is the **e** printing it.

***Graphical user interface, application, Word

Description automatically generated***

* *Let’s* changethe position of the input statement to find out what type of error it usually gives and as per the below screenshot it gives value error which is an exception so we can add it as exception
* *Graphical user interface, text, application, email

  Description automatically generated*
* Let’s mention the value error and zerodivision error specifically in the except blocks and if we don’t know the other errors we can generally mention as exception and we can print a general error message as something went wrong. Move the input inside the try block and try to replicate zerodivision error and value errors and observe the output. Check the below code for more information.

Graphical user interface, text, application

Description automatically generated

a = 5

b = 2

try:

    print("resource opened")

    print(a/b)

    k = int(input("enter the number: "))

    print(k)

except ValueError as e:

    print('Invalid input or do not enter alphabets', e)

except ZeroDivisionError as e:

    print('a number cannot be divided by zero', e)

except Exception as e:

    print('Something went wrong')

finally:

    print("resource closed")

print("Bye")