17th Jun Assignment Python Baic

1. The try block is where we write the code intended to perform he desired operation in a program in case of any errors this block raises an exception. The exception block lets you handle the error raised in a try block.
2. Syntax:

try:

  #some code

except:

  #error handling code

1. In such a case error is reported by the system and remaining code fails to execute
2. In case of bare exception we don’t mention the name of the exception in front of except statement. An exception with a specific exception type can handle only that exception while a bare exception can handle any exception raised by try block
3. x = 10

y = 0

str1 = 'test'

try:

print("outer try block")

try:

print("nested try block")

print(x + str1 )

except TypeError as te:

print("nested except block")

print(te)

print(x / y)

except ZeroDivisionError as ze:

print("outer except block")

print(ze)

1. Yes . eg:

try:

    mylist = [1, 2, 3]

    mylist[3]

    my\_dict = {"name":"Shri Hari","power":"infinite"}

    print(my\_dict["Stay"])

    with open("my\_file.txt", "r") as myfile:

        content = myfile.read()

except KeyError:

    print("not a key")

except IndexError:

    print("tring to access an item above list size")

except FileNotFoundError:

    print("File is not available")

1. EOFError: it is raised when one of the built-in functions input() or raw\_input() hits an end-of-file condition (EOF) without reading any data.
2. FloatingPointError: It is raised by floating point operations that result in errors, when floating point exception control (fpectl) is turned on. Enabling fpectl requires an interpreter compiled with the --with-fpectl flag
3. IndexError: Raised when a sequence subscript is out of range.
4. MemoryError**:** Raised when an operation runs out of memory.
5. OverflowError**:** Raised when the result of an arithmetic operation is too large to be expressed by the normal number format.
6. TabError: It occurs due to inconsistent use of tabs and spaces in indentation, error is raised when you try to indent code using both spaces and tabs.
7. ValueError**:** Raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.
8. try:

    n = int(input("Please enter the numerator: "))

    d = int(input("Please enter the denominator: "))

    x = n/d

except ZeroDivisionError:

    print("div cant be done by zero")

else:

  print(x)

1. a = input('Enter a number')

try:

  n = int(a)

except TypeError:

  print('Wrong type of data entered')

else:

  print(n)

1. try:

mylist = [1, 2, 3]

mylist[3]

except IndexError:

print("tring to access an item above list size")

1. try:

my\_dict = {"name":"Shri Hari","power":"infinite"}

print(my\_dict["Stay"])

except KeyError:

print("not a key")

1. try:

my\_dict = {"name":"Shri Hari","power":"infinite"}

print(my\_dict["Stay"])

except:

print("There is some error")