In [4]:

```python
help(Exception)
```

```
Help on class Exception in module builtins:

class Exception(BaseException)
 |  Common base class for all non-exit exceptions.
 |
 |  Method resolution order:
 |      Exception
 |      BaseException
 |      object
 |
 |  Built-in subclasses:
 |      ArithmeticError
 |      AssertionError
 |      AttributeError
 |      BufferError
 |      ... and 15 other subclasses
 |
 |  Methods defined here:
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(...)
 |      Helper for pickle.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

In [8]:

```python
help(BaseException)
```

```
Help on class BaseException in module builtins:

class BaseException(object)
 |  Common base class for all exceptions
 |
 |  Built-in subclasses:
 |      Exception
 |      GeneratorExit
 |      KeyboardInterrupt
 |      SystemExit
 |
 |  Methods defined here:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __reduce__(...)
 |      Helper for pickle.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

In [6]:

```python
6/0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4364/2590173015.py in <module>
----> 1 6/0

ZeroDivisionError: division by zero
```

In [7]:

```python
help(ZeroDivisionError)
```

```
Help on class ZeroDivisionError in module builtins:

class ZeroDivisionError(ArithmeticError)
 |  Second argument to a division or modulo operation was zero.
 |
 |  Method resolution order:
 |      ZeroDivisionError
 |      ArithmeticError
 |      Exception
 |      BaseException
 |      object
 |
 |  Methods defined here:
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from BaseException:
 |
 |  __delattr__(self, name, /)
 |      Implement delattr(self, name).
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __reduce__(...)
 |      Helper for pickle.
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __setattr__(self, name, value, /)
 |      Implement setattr(self, name, value).
 |
 |  __setstate__(...)
 |
 |  __str__(self, /)
 |      Return str(self).
 |
 |  with_traceback(...)
 |      Exception.with_traceback(tb) --
 |      set self.__traceback__ to tb and return self.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from BaseException:
 |
 |  __cause__
 |      exception cause
 |
 |  __context__
 |      exception context
 |
 |  __dict__
 |
 |  __suppress_context__
 |
 |  __traceback__
 |
 |  args
```

In [11]:

```python
# IndexError
st="KKK"
for ch in range(0,len(st)+1):
    print(st[ch])
```

```
K
K
K
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4364/590709430.py in <module>
      2 st="KKK"
      3 for ch in range(0,len(st)+1):
----> 4     print(st[ch])

IndexError: string index out of range
```

In [12]:

```python
# TypeError
ans="dffhjhk"+56
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4364/997365479.py in <module>
      1 # TypeError
----> 2 ans="dffhjhk"+56

TypeError: can only concatenate str (not "int") to str
```

In [14]:

```python
# ValueError
num=int(input("Enter Number"))
print(num)
```

```
Enter Numberdhgfjf
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4364/3429185578.py in <module>
      1 # ValueError
----> 2 num=int(input("Enter Number"))
      3 print(num)

ValueError: invalid literal for int() with base 10: 'dhgfjf'
```

In [15]:

```python
# NameError
x=ans+7
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4364/1480440498.py in <module>
----> 1 x=ans+7

NameError: name 'ans' is not defined
```

In [20]:

```python
try:
    a=int(input("Number 1 :"))
    b=int(input("Number 2 :"))
    ans=a/b
    print("Division is :",ans)
except ZeroDivisionError as e:
    print("Don't divide bt ZERO")
except ValueError as v:
    print("Please enter numeric value!")
except:
    print("Exception occured")
```

```
Number 1 :3
Number 2 :0
Don't divide bt ZERO
```

In [22]:

```python
try:
    #a=int(input("Number 1 :"))
    b=int(input("Number 2 :"))
    print("x",x)
    ans=x/b
    print("Division is :",ans)
except ZeroDivisionError as e:
    print("Don't divide bt ZERO")
except ValueError as v:
    print("Please enter numeric value!")
except:
    print("Exception occured")
```

```
Number 2 :45
Exception occured
```

In [35]:

```python
l1=[3,4,5,6,7]
try:
    sum=0
    for i in l1:
        sum+=i
    #print(sum)
    av=sum/len(l1)
    #print("Average",av)
except TypeError as e:
    print(e)
    print("It is Type Error")
except NameError as e:
    print(e)
except ZerodivisionError as e:
    print(e)
else :
    print(sum)
    print("Average",av)
finally:
    print("Exception complete")
```

```
25
Average 5.0
Exception complete
```

In [41]:

```python
# AssertionError
try:
    age=10
    assert age>18, "Age is not above 18.....it the problem"
    print("Eligible for Votting")
except AssertionError as e:
    print(e)
else:
    print("Verification Done")
```

```
Age is not above 18.....it the problem
```

In [42]:

```python
def verify():
    age=10
    assert age>18, "Age is not above 18.....it the problem"
    print("Eligible for Votting")
try:
    verify()
except AssertionError as e:
    print(e)
else:
    print("Verification Done")
```

```
Age is not above 18.....it the problem
```

In [5]:

```python
import sys
def platformBasedTask(color):
    assert "linux" in sys.platform, "This is not supported them on this platform"
    print("Theme set to",color)
    print(color,"Color theme is supported",sys.platform)
try:
    platformBasedTask("BLUE")
except AssertionError as e:
    print(e)
except:
    print("Something went wrong")
```

```
This is not supported them on this platform
```

In [ ]:

```python
# Constructor Overloading
```

In [11]:

```python
class InvalidAgeException(Exception):
    def __int__(self,err="Invalid age"):
        self.err=err
    def showError(self):
        print(err)
```

In [14]:

```python
try:
    age=int(input("Enter your Age"))
    if(age<0 and age>18):
        raise InvalidAgeException("Please enter the number")
except InvalidAgeException as e:
    e.showError()
else:
    print("Login successfully!")
```

```
Enter your Age21
Login successfully!
```

In [1]:

```python
# File Handling
f=open("food.py")
#print(f.read())
#print(f.read(8))
#print(f.readline())
print(f.readlines())
f.close()
```

```
['from FoodWorld import food\n', '\n', 'print("This is food world")\n', 'print("Module is",__name__)\n', '\n', "v=food('Pizza',3,540)\n", 'v.display()\n', 'v.calcBill()\n', 'print("_______________")\n']
```

In [2]:

```python
try:
    f=open("food.py")
    print(f.readlines())
except FileNotFoundError as e:
    print(e)
else:
    print("Reading done!")
finally:
    f.close()
```

```
['from FoodWorld import food\n', '\n', 'print("This is food world")\n', 'print("Module is",__name__)\n', '\n', "v=food('Pizza',3,540)\n", 'v.display()\n', 'v.calcBill()\n', 'print("_______________")\n']
Reading done!
```

In [3]:

```python
try:
    with open("food.py") as f:
        print(f.readlines())
except FileNotFoundError as e:
    print(e)
else:
    print("Reading done!")
```

```
['from FoodWorld import food\n', '\n', 'print("This is food world")\n', 'print("Module is",__name__)\n', '\n', "v=food('Pizza',3,540)\n", 'v.display()\n', 'v.calcBill()\n', 'print("_______________")\n']
Reading done!
```

In [4]:

```python
try:
    with open("food.py","w") as f:
        f.write("Good Evening!")
except FileNotFoundError as e:
    print(e)
else:
    print("Reading Done!")
```

```
Reading Done!
```

In [12]:

```python
try:
    with open("School.py","a") as f:
        f.write("Let's complete File Handeling today!")
    with open("School.py","r") as f1:
        print(f1.read())
except FileNotFoundError as e:
    print(e)
else:
    print("Writing and Reading Done!")
```

```
Let's complete File Handeling today!
Writing and Reading Done!
```

In [6]:

```python
try:
    with open("Calculator.py",'r') as f:
        with open("food.py",'a') as f1:
            for i in f:
                f1.write(i)
        print(type(f1))
except FileNotFoundError as e:
    print(e)
else:
    print("Reading and Writing Done!")
```

```
<class '_io.TextIOWrapper'>
Reading and Writing Done!
```

In [ ]:

```python
# Assignment
```

In [17]:

```python
# 1.Read contents of file character by character.
try:
    with open('food.py') as f:
        for i in f:
            for ch in i:
                print(ch,end=' ')
except FileNotFoundError as e:
    print(e)
else:
    print("Reading Done!")
```

```
r e g h f r o m   F o o d W o r l d   i m p o r t   f o o d

 p r i n t ( " T h i s   i s   f o o d   w o r l d " )
 p r i n t ( " M o d u l e   i s " , _ _ n a m e _ _ )

 v = f o o d ( ' P i z z a ' , 3 , 5 4 0 )
 v . d i s p l a y ( )
 v . c a l c B i l l ( )
 p r i n t ( " _ _ _ _ _ _ _ _ _ _ _ _ _ " )
 d f g L e t ' s   c o m p l e t e   F i l e   H a n d e l i n g   t o d a y ! Reading Done!
```

In [10]:

```python
# 2.Read contents of file line by line.
try:
    filename = "food.py"
    with open(filename) as diary_file:
        n = 1
        for line in diary_file:
            print(n, line)
            n += 1
except FileNotFoundError as e:
    print(e)
else:
    print("Reading Done!")
```

```
1 from FoodWorld import food

2

3 print("This is food world")

4 print("Module is",__name__)

5

6 v=food('Pizza',3,540)

7 v.display()

8 v.calcBill()

9 print("_____________")

Reading Done!
```

In [11]:

```python
# 3.Read contents of file word by word.
try:
    with open("food.py",'r') as file:
        for line in file:
            for word in line.split():
                print(word)
except FileNotFoundError as e:
    print(e)
else:
    print("Reading Done!")
```

```
from
FoodWorld
import
food
print("This
is
food
world")
print("Module
is",__name__)
v=food('Pizza',3,540)
v.display()
v.calcBill()
print("_____________")
Reading Done!
```

In [13]:

```python
# 4.remove file 1
import os
if os.path.exists("food.py"):
    os.remove("food.py")
    print("Removed")
else:
    print("The file does not exist")
```

```
Removed
```

In [15]:

```python
# 5.Copy contents from one file to another
try:
    with open('Calculator.py','r') as f, open('food.py','a') as f1:
        for i in f:
            f1.write(i)
except FileNotFoundError as e:
    print(e)
else:
    print("Copy Done!")
```

```
Copy Done!
```

In [ ]: