PYTHON

From Simple to Complex
With
Examples

AYESHA NOREEN

Bachelor's in Software Engineering,

Master's in Computer Science

from COMSATS University, Islamabad

NOTE!!!

In these notes Screenshots of practice examples and coding are added. The code files are also available in code folder that contain .ipynb files that are created on Jupyter notebook.

Chapter19
Some Common errors in Python

- **Syntax Error**

Syntax error occur due to invalid syntax e.g.

```
def fun1():
    print('Hello') #valid syntax so no error
def fun2
    pass #invalid syntax miss paramthesis and colon
```

```
  Cell In[4], line 3
    def fun2
            ^
SyntaxError: invalid syntax
```

- **Name Error**

name error occur due to invalid variable name or misspelled variable name or undeclared variable name e.g.

```
name="Ayesha" #we declare a variable name
print(age) #we want to print age which is not declared so it prints
#NameError: name 'age' is not defined
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 2
      1 name="Ayesha" #we declare a variable name
----> 2 print(age)

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

- **Type Error**

type error occur due to invalid data type e.g.

```
a=5 #declare an int
str="Ayesha" #declare a string
print(str+a) #want to concate int and string so give type error and prints
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 3
      1 a=5 #declare an int
      2 str="Ayesha" #declare a string
----> 3 print(str+a)

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

- **Index Error**

index error occur due to invalid index number e.g.

```
l=[1,2,3,4,5] #we declare a list whose maximum index is 4
print(l[5]) #if we want to print index 5 value which is not present
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[7], line 2
      1 l=[1,2,3,4,5] #we declare a list whose maximum index is 4
----> 2 print(l[5])

IndexError: list index out of range
```

- **Indentation Error**

Indentation mean number of spaces. Indentation error occur due to wrong indentation or wrong order of indentation e.g.

```
def func(): #here we define a function
print("Hello") #here we disturb its spaces
print("world")
func() #now when we call function it gives error
```

```
  Cell In[8], line 2
    print("Hello") #here we disturb its spaces
    ^
IndentationError: expected an indented block after function definition on line 1
```

- **value Error**

Value error occur if we want to convert a non convertible data type into another data type e.g.

```
string='123'
print(int(string)) #no error it convert string into int and prints 123
str='abc'
print(int(str)) #now it give value error bcz string value is not convertable into int
```

```
123


---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 4
      2 print(int(string)) #no error it convert string into int and prints 123
      3 str='abc'
----> 4 print(int(str))

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

- **attribute Error**

attribute error occur if we try apply an operation which is not pre-defined e.g.

```
l=[1,2,3,4,5] #here we declare a list
l.push(6) #and apply a push method which is not applicable on list
print(l) #now if we print list it gives error as
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[10], line 2
      1 l=[1,2,3,4,5] #here we declare a list
----> 2 l.push(6) #and apply a push method which is not applicable on list
      3 print(l)

AttributeError: 'list' object has no attribute 'push'
```

- **key Error**

key error occur when we try to access a key from a dictionary which is not present. e.g.

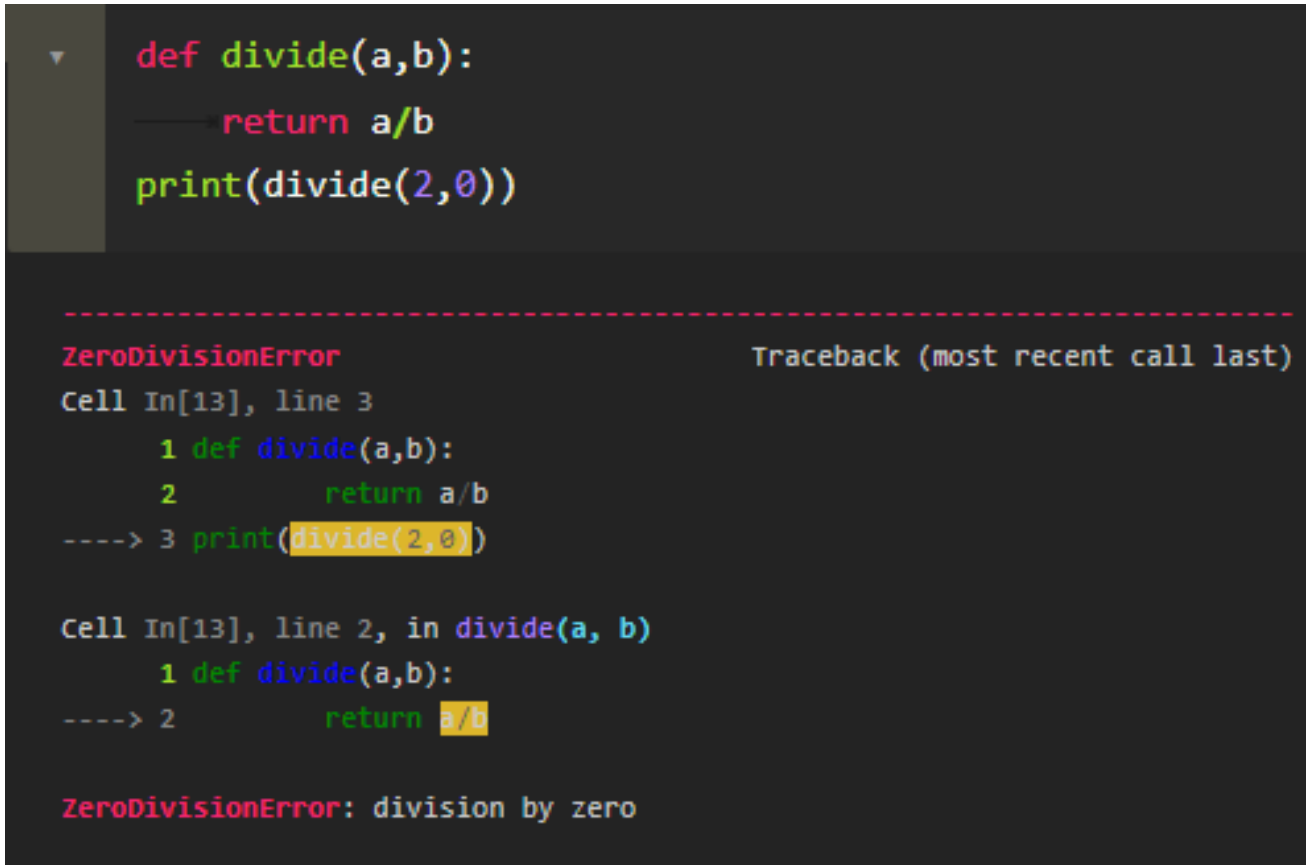
```
d={'name':'Ayesha'} #declare a dictionary of key name
print(d['age']) #we want to print key age which is not present
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[11], line 2
      1 d={'name':'Ayesha'} #declare a dictionary of key name
----> 2 print(d['age'])

KeyError: 'age'
```

- **zeroDivisionError**

ZeroDivisionError occur when we try to divide a number by zero. e.g.



```
def divide(a,b):
    return a/b
print(divide(2,0))
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[13], line 3
      1 def divide(a,b):
      2         return a/b
----> 3 print(divide(2,0))

Cell In[13], line 2, in divide(a, b)
      1 def divide(a,b):
----> 2         return a/b

ZeroDivisionError: division by zero
```

• notimplementedError

Not implemented error raise when we want to print something that is not present.

```
class Student:
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def reg_no(self):
        raise NotImplementedError("You have to define reg_no method for every class")
class Adnan(Student):
    def __init__(self,name,age):
        self.name=name
        self.age=age
a=Adnan('Ali','5')
a.reg_no()
```

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
Cell In[31], line 12
     10         self.age=age
     11 a=Adnan('Ali','5')
---> 12 a.reg_no()

Cell In[31], line 6, in Student.reg_no(self)
      5 def reg_no(self):
----> 6     raise NotImplementedError("You have to define reg_no method for every class")

NotImplementedError: You have to define reg_no method for every class
```

- # Implement errors by yourself

```
def add(a,b):
    return a+b
add(2,3)
add('2','3')
#here we want to add two numbers but if we pass string of numbers than it concate them
#not add we can solve this problem as
def add1(a,b):
    if (type(a)==int and type(b)==int):
        return a+b
    else:
        return "you enter wrong data type"
#now this function add two numbers if these are integers otherwise return a string which show
#error msg but this is not good bcz if we want after adding numbers the function return sum which
#is store in database but this function always return something either addition or string and than
#string also stored in database in this case but we want to raise error msg instread we string as
def add2(a,b):
    if (type(a)==int and type(b)==int):
        return a+b
    else:
        raise TypeError("you enter wrong data type") #here define details of error
```

- # Implement errors by yourself

```
print(add(2,3)) #it prints 5
print(add('2','3')) #it prints 23
print(add1(2,3)) #it prints 5
print(add1('2','3')) #it prints you enter wrong data type
print(add2(2,3)) #it prints 5
print(add2('2','3')) #it prints TypeError: you enter wrong data type
```

```
5
23
5
you enter wrong data type
5


---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[21], line 27
     25 print(add1('2','3')) #it prints you enter wrong data type
     26 print(add2(2,3)) #it prints 5
---> 27 print(add2('2','3'))

Cell In[21], line 20, in add2(a, b)
     18     return a+b
     19 else:
---> 20     raise TypeError("you enter wrong data type")

TypeError: you enter wrong data type
```

- **Exception handling**
  - **Try:**

In try block there may be one or more than one statements and it is usually used to check condition.

  - **Except:**

Except block is used to throws exception we can use more than one blocks of except.

  - **Else**

Else block executes if there is no exception.

  - **Finally**

Finally block always executes.

Example

```
while True:
    try:
        number=int(input("Enter any number:"))
    except ValueError:
        print("you d'nt enter number!!")
    except:
        print("unexpected error! ! !")
    else:
        print(f"Number is:{number}")
        break
    finally:
        print("I am finally block.....")
```

```
Enter any number:u
you d'nt enter number!!
I am finally block.....
Enter any number:i
you d'nt enter number!!
I am finally block.....
Enter any number:)
you d'nt enter number!!
I am finally block.....
Enter any number:5
Number is:5
I am finally block.....
```

• TODO Task

Define a function that return division of two numbers if number is divided by zero than throws an exception ZeroDivisionError and if number is divided by anything else other than integer and float than if throws a type error.as

```
def divide(a,b):
    try:
        return a/b
    except ZeroDivisionError:
        print("number is not divided by zero")
    except TypeError:
        print("you have to enter integers and floats only")
    except:
        print('Unexpected error !!!')
print(divide(2,2))
```

```
1.0
```

• Custom Errors

When we want to print errors according to our own need that we define class of our own defined error first than inherit it from any pre defined error than use it as

```
class TooShortLengthError(ValueError):
    pass
def validate(name):
    if len(name)<8:
        raise TooShortLengthError("length name is too short")
    else:
        print(f"Your name is:{name}")
n=input("Enter any name:")
print(validate(n))
```

```
Enter any name:ali


---------------------------------------------------------------------------
TooShortLengthError                       Traceback (most recent call last)
Cell In[36], line 17
     15         print(f"Your name is:{name}")
     16 n=input("Enter any name:")
---> 17 print(validate(n))

Cell In[36], line 13, in validate(name)
     11 def validate(name):
     12     if len(name)<8:
---> 13         raise TooShortLengthError("length name is too short")
     14     else:
     15         print(f"Your name is:{name}")

TooShortLengthError: length name is too short
```

- **Module**

Module is a file that contain classes and functions wrote by developer.

- **Pdb module**

In python for debugging we have to import pdb module. pdb stands python debugger.

- **Debugging**

Debugging is a process of finding and resolving defects and errors of a computer program that prevent correct operation of a computer system or software.we use debugging when our program is not working properly or throws some exception and also we use it when our program is not working properly as we want.we can do debugging in two steps

1)set trace

For set trace we have to import pdb python debugger.

2)executes code line by line

Than debugger show →on line at which debugger is working

Than we have to place n command for moving or debugging to n

- # Example

```
import pdb
pdb.set_trace()
name=input("Enter you name:")
age=int(input("Enter you age:"))
print(f"Hello!!! {name} your age is {age}")
age2=age+5
print(f"{name}, After 5 years you will be :{age2} years old")
```

```
--Return--
None
> c:\windows\temp\ipykernel_9620\583253884.py(2)<module>()

ipdb> c
Enter you name:ayesha
Enter you age:23
Hello!!! ayesha your age is 23
ayesha, After 5 years you will be :28 years old
```

• Commands

There are following commands for pdb:

- N normally executes
- L show list
- Q quit
- C continue execution without line by line execution