

Python Handwritten Notes

Exception Handling

* Exception :- An exception is an event, which occurs during the execution of program that disrupts the normal flow of the program.

→ There exists some errors which occur at run time in the program.

for instance, attempting to divide by zero, accessing a list which is not defined, opening a file that does not exist are common example of run time errors.

The run time error is called exception.

Common Exceptions:-

1. ZeroDivisionError :- Occurs when a number is divided by zero.
2. NameError :- It occurs when a name is not found. It may be local or global.
3. Indentation Error :- If incorrect indentation is given.

Example:-

```
marks = 1000
```

```
a = marks / 0  
print (a)
```

Output:-

```
a = marks / 0
```

```
ZeroDivisionError: division by zero
```

* Difference between Syntax Error and Exceptions:-

- Syntax Error:- This error is caused by the wrong syntax in the code.

Example:-

```
amount = 100
```

```
if (amount > 299)
```

```
print ("you are eligible to purchase")
```

Output:-

invalid syntax

- Exceptions:- An exception can be defined as an unusual condition in program.

* Python Built in Exceptions:- Exceptions which are available in Python language.

The built-in exceptions can be generated by the interpreter or built in functions.

There are several built-in exceptions in Python that are raised when error occurs.

Some Built in Exceptions:-

Exception	Description
ArithmeticError	Raised when an error occurs in numeric calculations.
ImportError	Raised when an imported module does not exist.
IndentationError	Raised when indentation is not correct.
AttributeError	Raised when attribute assignment or reference fails.
IndexError	Raised when an index of a sequence does not exist.
SyntaxError	Raised when a syntax error occurs.
ValueError	When there is a wrong value in a ^{specified} data type.
ZeroDivisionError	When the second operand in a division is zero.

Exception Handling:-

In Python, exceptions can be handled using a `try` and `except` statements

→ Try and except statements are used to catch and handle exceptions in Python.

(Try and Except)

→ Statements that can raise exceptions are kept inside the `try` clause and the statements that handle the exception are written inside `except` clause.

```
try { Run this code }
```

```
except { Run this code if an exception occurs }
```

Syntax:-

```
try:  
    # block of code  
except Exception1:  
    # block of code
```

Example 1:-

```
try:
    a = int(input("Enter a:"))
    b = 0
    c = a/b
except:
    print("Can't divide with zero")
```

Output:-

Enter a : 10
Can't divide with zero

Example (2):-

```
a = [1, 2, 3]
try:
    print("Second element is ", (a[1]))
    # Throws error, there are only 3 elements
    print("Fourth element is ", (a[3]))
except:
    print("An error occurred")
```

Output:-

Second element 2
An error occurred

● Catching Specific Exceptions in Python:-

→ We can also handle exceptions separately rather than all exceptions in one block.

Syntax:-

```
try :  
    # statement  
except IndexError :  
    # handle IndexError  
except ValueError :  
    # handle ValueError
```

for example:-

```
def fun (a) :  
    if a < 4 :  
        b = a / (a - 3)  
  
        print("value of b = ", b)  
  
    try :  
        fun(3)  
        fun(5)  
    except ZeroDivisionError :  
        print("Zero Division Error Occurred  
and Handled")
```

```
except NameError:  
    print("NameError Occurred and Handled")
```

Output:-

Zero DivisionError Occurred and Handled

* Finally block:-

- Python provides the optional finally statement which is used with the try statement.
- The finally block provides a guarantee of the execution.

Syntax:-

```
try:  
    # block of code  
    # throws an exception
```

```
finally:  
    # block of code  
    # this will always be executed
```

`try` { `Run this code` }

`except` { `Run this code if an exception occurs` }

`finally` { `Always run this code` }

Example:-

```
try:  
    a = 5/0  
    print(a)  
except ZeroDivisionError:  
    print("can't divide by zero")  
finally:  
    print("This is always executed")
```

Output:-

can't divide by zero
This is always executed

* Python User-defined Exceptions:-

- In Python, user can create their own exceptions. This can be done by creating a new class which is derived from the exception class.
- A Programmer can create his own exceptions, called user-defined exceptions or custom Exception.
 - Creating Exception class using Exception class
 - Raising Exception
 - Handling Exception.

1. Creating Exception:-

Syntax:-

```
class MyException (Exception):  
    pass
```

- ### 2. Raising Exception:-
- raise statement is used to raise the user-defined Exception.

Syntax:-

raise MyException("message")

3. Handling Exception:-

Using try and except block Programmer can handle exceptions.

Example:

```
class MyException(Exception):  
    pass  
  
c = 25  
if c > 5:  
    raise MyException("Something Went Wrong")
```

Output:

MyException: Something Went Wrong

Notes by Ipwebdevelopers