

## Assignment 07 Solutions

### Q1. What is the purpose of the try statement?

**Ans:** `try` statement is used for exception handling in python. It consists of a block of risky statements which might cause an exception during runtime. If code within try block raises exception, then the exception will be reverted to the corresponding except block when multiple except blocks are present, else it will be reverted to the default except block. In short, try and except in union avoid programs from crashing during runtime due to exceptions.

In [1]:

```
1 try:
2     print(30/0)
3 except ZeroDivisionError:
4     print('Division by Zero is Not Possible')
```

Division by Zero is Not Possible

### Q2. What are the two most popular try statement variations?

**Ans:** The Popular try statement variations are:

1. `try, except`
2. `try, except, finally`
3. `try, except, finally, else`

The `try` block is used to check code for exceptions during runtime. i.e. code inside `try` block will execute completely when there is no error in the program where as the code inside `except` block will execute whenever the program encounters some error in the preceeding `try` block.

Where as the code enters the `else` block if and only if the try clause does not raise an exception. The code in the `finally` block will execute irrespective of exception.

### Q3. What is the purpose of the raise statement?

**Ans:** `raise` statement is used to trigger an exception explicitly, if a certain condition is not as per requirement of programmer. `raise` statement helps in triggering exception as per programming logic.

In [2]:

```
1 #raising an exception
2 input = input("Enter the input:")
3 if not type(input) is int:
4     raise TypeError("Only integers are allowed") #exception will be thrown
```

Enter the input:abhishek

```
-----
TypeError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_23800\3107196368.py in <module>
      2 input = input("Enter the input:")
      3 if not type(input) is int:
----> 4     raise TypeError("Only integers are allowed") #exception will be thrown
```

**TypeError:** Only integers are allowed

### Q4. What does the assert statement do, and what other statement is it like?

**Ans:** There are few assertions that programmer always want to be true to avoid code failure. This type of requirement is fulfilled by `assert` statement. This statement takes a boolean condition output of which is True which in turn helps Further Program Execute. If output of assert statement is False, it raises an Assertion Error.

In [3]:

```
1 assert(2 == 10), f'this is not a valid statement'
```

```
-----
AssertionError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_23800\4203093858.py in <module>
----> 1 assert(2 == 10), f'this is not a valid statement'
```

**AssertionError:** this is not a valid statement

**Q5. What is the purpose of the with/as argument, and what other statement is it like?**

**Ans:** **with/as** statement simplifies use of file handling in python. When we use a **with** statement for file reading, there is no need for programmer to explicitly take care of activities like resource deallocation and file closing by using `file.close()` method. **with** statement itself ensures proper acquisition and release of resources. This avoids triggering of exceptions if the file closing is unknowingly forgotten in the code execution.

```
with open('sample_file.txt', 'w') as file:  
    file.write('Hello World')
```