# 1) What is an exception in Python?

```
# An Exception is a case where the python code errors out due to logical errors rather then its syntax.
```

# 2) What is the purpose of using exception handling in Python?

```
# It is to catch any logical errors during program runtime.
```

# 3) What is the difference between a syntax error and an exception in Python?

```
# A Syntax error is one where there is an error in the syntax of the python code where as an exception is an error thats occured after the program has run and errors out on other factors that aren't exactly the program's fault.
```

# 4) How do you raise an exception in Python?

```
# By Using the Raise keyword.
```

# 5) What is the try-except block in Python?

```
# Try except is snytax in python that is designed to catch forseen exception that may or may not occur during the execution of the program.
```

#6) What is the syntax of the try-except block in Python?

```
# try < code_snippet > except
```

# PART 2 #1) Write a Python program that prompts the user to enter two integers and then divides the first integer by the second integer, Handle any exceptions that may occur during the division operation.

```
a,b = map(int,input("Enter 2 numbers: ").split())
try:
    result = a/b
    print(result)
except Exception as e:
    print("An error occured: ",e)

Enter 2 numbers: 0 0
An error occured:  division by zero
```

#2) Write a Python program that reads a file name from the user and attempts to open the file, Handle any exceptions that may occur if the file does not exist or if there is an error reading the file.

```
try:
    with open(input("Enter file name: "),'r') as file:
        pass
except Exception as e:
    print("An error occured: ",e)

Enter file name: afsdafsd
An error occured:  [Errno 2] No such file or directory: 'afsdafsd'
```

#3) Write a Python program that prompts the user to enter a list of numbers and calculates the average of those numbers, Handle any exceptions that may occur if the user enters non-numeric values.

```
try:
    userins = list(map(int,input("Enter a list of numbers: ").split()))
    print((sum(userins)/len(userins)))
except Exception as e:
    print("Some error occured: ",e)

Enter a list of numbers: 1 2 3 4 5
3.0
```

# 4) Write a Python program that generates a random number between 1 and 10 and prompts the user to guess the number, Handle any exceptions that may occur if the user enters non-numeric values or if the guess is out of range.

```
import random
random_num = random.randint(1,10)
print(random_num)
try:
    user_guess = int(input("Enter your guess: "))
    if user_guess < 1 or user_guess > 10:
        raise IndexError("Your Guess was too high or too low. Guess a num between 1 and 10")
    if user_guess == random_num:
        print("You guessed Correct")
    else:
        print("You guessed wrong.")
except ValueError:
    print("Invalid input, enter a number")
except IndexError as e:
    print(e)


9
Enter your guess: a
Invalid input, enter a number
```

# 5) Write a Python program that prompts the user to enter a list of integers and then calculates the sum of those integers, Handle any exceptions that may occur if the user enters non-numeric values.

```
try:
    userins = map(int,input("Enter numbers: ").split())
    print(sum(userins))
except ValueError:
    print("Enter a number not a letter")

Enter numbers: abcdef ghi
Enter a number not a letter
```

#6) Using try…except, showcase the ZeroDivisionError.

```
b,a=0,3
try:
    print(a/b)
except ZeroDivisionError:
    print("Cant divide by 0")


Cant divide by 0
```

# 7) Create a simple list containing five elements and try to print the sixth element of the list, [use IndexError exception].

```
elements = [i for i in range(5)]
try:
    print(elements[5])
except IndexError:
    print("Array contains only elements upto index 4")

Array contains only elements upto index 4
```

# 8) Try printing a variable without declaring it, [use NameError exception]:

```
a=9
try:
    print(a)
except NameError:
    print("variable isnt defined yet")


9
```

# 10) In Python, we can choose to throw an exception if a condition occurs, To throw an exception, we use the ‘raise’ keyword, Show an example.

```
a=9
try:
   a+=1
except NameError:
    print("variable isnt defined yet")
else:
    print("Works")

Works
```

# Part 3 # 1) Write a Python program that prompts the user to enter a string and then converts it to an integer, Handle the ValueError exception that may occur if the string cannot be converted to an integer.

```
try:
    list = input("Enter a string: ")
    int_list = [ord(x) for x in list]
    print(int_list)
except ValueError as e:
    print("Cannt convert into string",e)

Enter a string: aaa@@@@
[97, 97, 97, 64, 64, 64, 64]
```

# 2) Ask the user for the numerator and denominator value; and perform division, If the user enters a number, the program will evaluate and produce the result.

```
a,b = (input("Enter 2 numbers: ").split())
try:
    print(int(a)/int(b))
except ValueError as e:
    print(e)
except ZeroDivisionError:
    print("Cant divide by 0")


Enter 2 numbers: 2 3
0.6666666666666666
```

# 3) Ask the user to enter an amount of money, In the try block, run a condition to check if the input value is less than 10 thousand; in which case raise a ValueError and print your message inside it, In the except block, catch the ValueError we previously raised and print the message inside it.

```
try:
    userin = int(input("Enter money: "))
    if userin < 10000:
        raise ValueError("Not Enough Money")
    print("You Gave me: ",userin)
except ValueError as e:
    print(e)



Enter money: 69
Not Enough Money
```

# 4) Write a Python function that takes a list of integers as an argument and returns the largest integer in the list, Handle the ValueError that may occur if the argument is an empty list.

```
def largest(args):
    try:
        if len(args) == 0:
            raise IndexError("List Is Empty")
        print(max(args))
    except IndexError as e:
        print(e)

largest([1,2,3,4,5,6,7,8])


8
```

# 5) Write a Python function that takes a dictionary as an argument and returns the value associated with a given key, Handle the KeyError that may occur if the key does not exist in the dictionary.

```
def return_value(dict,key):
    try:
        return dict[key]
    except KeyError:
        return "Key Doesnt exist"
print(return_value({1:2,2:3,3:4},6))

Key Doesnt exist
```

# 6) Write a Python function that takes two strings as arguments and concatenates them, Handle the TypeError that may occur if either argument is not a string.

```
def two_strings(str1,str2):
    try:
        return str1+str2
    except TypeError:
        return "Arguments both arent string"
print(two_strings('a',4))


Arguments both arent string
```

#Part 4 #You are working on a program that keeps track of the grades for a class of students, The program should allow the user to input the name and grades of each student, and then calculate and display the average grade for the class.

```
student_names = []
student_grades = {}
for i in range(1,int(input("Enter the number of students in the class: "))+1):
    while True:
        try:
            student_name,grade = input("Enter student's name and their grade seperated by comma:
").split(',')
            if int(grade) < 0 or int(grade) > 100 or not grade.isdigit():
                print("Invalid Grade, Please Enter Again")
                continue
                grade = float(grade)
        except:
            print("Some error occured")
        student_names.append(student_name)
        student_grades[student_name] = int(grade)
        break
print(".join(format(student_grades[student_name],'.2f') + "    " for student_name in
student_grades))
```

```
Enter the number of students in the class: 2
Enter student's name and their grade seperated by comma: A,2
Enter student's name and their grade seperated by comma: B,3
2.00    3.00
```