1 Generate Atleast 5 different Errors

```
prnt("Hello")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-7355f84c22cf> in <module>
----> 1 prnt("Hello")

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

SEARCH STACK OVERFLOW

```
a = "5"
b = 3
c = a + b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-164a21d5d6ff> in <module>
      1 a = "5"
      2 b = 3
----> 3 c = a + b

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

SEARCH STACK OVERFLOW

```
numbers = [1, 2, 3]
print(numbers[3])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-7-5c7bfb8e463f> in <module>
      1 numbers = [1, 2, 3]
----> 2 print(numbers[3])

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

```
age = "twelve"
int(age)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-8-515a8d64c693> in <module>
      1 age = "twelve"
----> 2 int(age)

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

SEARCH STACK OVERFLOW

```
a = 5/0
a
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-10-541f1ef54776> in <module>
----> 1 a = 5/0
      2 a

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

2. Handle all the 5 different Erros using try-except.

```
try:
  prnt("Hello")
except NameError:
  print("There is a name error")
```

```
There is a name error
```

```
try:
  a = "5"
  b = 3
  c = a + b
except TypeError:
  print("There is a type error")
```

```
There is a type error
```

```
try:
  numbers = [1, 2, 3]
  print(numbers[3])
except IndexError:
  print("There is a index error")
```

```
There is a index error
```

```
try:
  age = "twelve"
  int(age)
except ValueError:
  print("There is a value error")
```

```
There is a value error
```

```
try:
  a = 5/0
  a
except ZeroDivisionError:
  print("Cannot divide by zero!")
```

```
Cannot divide by zero!
```

3. Handle an error with try-except-else.

```
def divide(a,b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Cannot divide by zero!")
    else:
        print(f"The result is {result}.")

divide(4, 2)

divide(4, 0)
```

```
The result is 2.0.
Cannot divide by zero!
```

4. Handle an error with try-except-else-finally.

```
def divide(a,b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Cannot divide by zero!")
    else:
        print(f"The result is {result}.")
    finally:
        print("Division attempt complete.")


divide(4, 2)

divide(4, 0)
```

```
The result is 2.0.
Division attempt complete.
Cannot divide by zero!
Division attempt complete.
```

5. Use raise for generating User Defined Exception for minimum length of a list should be 5.

```
def check_list_length(lst):
    if len(lst) < 5:
        raise ValueError("The list must have at least 5 elements.")

try:
    check_list_length([1, 2, 3, 4])
except ValueError as e:
    print(e)
```

```
 The list must have at least 5 elements.
```

6 Create a file 'mod.py' with a class with multiple methods and few member variables. Also create an individual methods outside the class as well. Create another file 'test.py' and without executing the 'mod.py' get it executed using the 'test.py' file

```
#mod.py
class MyClass:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def add(self):
        return self.x + self.y

    def subtract(self):
        return self.x - self.y

def greeting(name):
    return f"Hello, {name}!"
```

```
#test.py
import mod

obj = mod.MyClass(3, 4)

print(obj.add())
print(obj.subtract())

print(mod.greeting("World"))
```

7 Using the above mentioned files 'test.py' and 'mod.py'. In 'test.py' create an object of the class defined in the 'mod.py'. Call the methods using the object. Make sure only that class is accessible and not the individual method to execute.

```
#test.py
from mod import MyClass

obj = MyClass(3, 4)

print(obj.add())
print(obj.subtract())
```

8 Using the files 'test.py' and 'mod.py' call the individual method which is defined outside the class in 'mod.py' and call in it test.py. Make sure only that method from the file 'mod.py' is accessible in 'test.py'.

```
#test.py
from mod import greeting

print(greeting("World"))
```

9. Using the files execute both class methods and individual method outside the class.

```
#test.py
from mod import MyClass, greeting

obj = MyClass(3, 4)

print(obj.add())
print(obj.subtract())
```

```
print(greeting("World"))
```

10. Create a script/program to open a file to write a string. Write a string in a file 'test_file.txt'.

```
with open('test_file.txt', 'w') as f:
    f.write("This is a test string")
```

11. Create a script/program to open a file 'test_file.txt' to read a string. Read the whole string content from the file and print it.

```
with open('test_file.txt', 'r') as f:
    content = f.read()
print(content)
```

12. Create a script/program to open a file 'test_file.txt' to read the content line by line and print it.

```
with open('test_file.txt', 'r') as f:
    for line in f:
        print(line)
```

13. Create a script/program to open a file 'test_file.txt' to append a string at the end of the existing string in a file.

```
with open('test_file.txt', 'a') as f:
    f.write("\nThis is a new string")
```

14. Create a script/program to write and read binary data in a file 'test_file.data'.

```
with open('test_file.data', 'wb') as f:
    f.write(b'\x01\x02\x03\x04')

with open('test_file.data', 'rb') as f:
    content = f.read()

print(content)
```

15. Using pickle dump no of variables with different data types in a file 'my_variables.data'

```
import pickle

a = "Hello"
b = [1, 2, 3]
c = {"key": "value"}

variables = [a, b, c]

with open('my_variables.data', 'wb') as f:
    pickle.dump(variables, f)
```

16. Create another script/program and read the dumped variables in the file 'my_variables.data'.

```
import pickle

with open('my_variables.data', 'rb') as f:
    variables = pickle.load(f)

print(variables)
```

17.Print the current date using datetime and date libraries.

```
import datetime

today = datetime.date.today()
```

```
print(today)
```

```
2022-12-22
```

18. Convert a datetime to a string.

```
from datetime import datetime

dt = datetime.now()
date_string = dt.strftime("%Y-%m-%d %H:%M:%S")
print(date_string)
```

```
2022-12-22 12:23:13
```

19.Get the difference between two dates in days.

```
from datetime import datetime, timedelta

date1 = datetime.strptime("2022-12-20", "%Y-%m-%d")
date2 = datetime.strptime("2022-12-25", "%Y-%m-%d")

difference = date2 - date1

print(difference.days)
```

```
5
```

20. Calculate your age in years, months and days.

```
from datetime import datetime, timedelta

birth_date = datetime.strptime("2000-11-07", "%Y-%m-%d")

today = datetime.today()

difference = today - birth_date

years = difference.days // 365
months = difference.days % 365 // 30
days = difference.days % 365 % 30

print(f"You are {years} years, {months} months, and {days} days old.")
```

```
You are 22 years, 1 months, and 20 days old.
```

21. Get the date which is 1 week from today's date.

```
from datetime import datetime, timedelta

# get the current date
today = datetime.today()

# calculate the date that is 1 week from today
one_week_from_today = today + timedelta(weeks=1)

print(one_week_from_today)
```

```
2022-12-29 12:30:42.706039
```

22. Get the date which is 1 year from today's date.

```
from datetime import datetime, timedelta

today = datetime.today()

one_year_from_today = today + timedelta(weeks=52)

print(one_year_from_today)
```

```
2023-12-21 12:31:50.582415
```

23. Get the date which is 1 month from today's date.

```
from datetime import datetime, timedelta

today = datetime.today()
one_month_from_today = today + timedelta(weeks=4)
print(one_month_from_today)
```

```
2023-01-19 12:32:58.988723
```

24. Get the 1st day of the current month from today's date.

```
from datetime import datetime

today = datetime.today()

first_day_of_month = today.replace(day=1)
print(first_day_of_month)
```

```
2022-12-01 12:34:26.803241
```

25. Get the 1st month of the current year from today's date.

```
from datetime import datetime

today = datetime.today()
first_month_of_year = today.replace(month=1)
print(first_month_of_year)
```

```
2022-01-22 12:38:21.703171
```

26. Get the dates of current month starting from Monday to Sunday in a list.

```
import calendar
import datetime

current_year = datetime.datetime.now().year
current_month = datetime.datetime.now().month
num_days_in_month = calendar.monthrange(current_year, current_month)[1]
dates = [datetime.date(current_year, current_month, day) for day in range(1, num_days_in_month+1)]
first_monday = next(i for i,d in enumerate(dates) if d.weekday() == calendar.MONDAY)
monday_to_sunday = dates[first_monday:]
print(monday_to_sunday)
```

```
[datetime.date(2022, 12, 5), datetime.date(2022, 12, 6), datetime.date(2022, 12, 7), datetime.date(2022, 12, 8), datetim
```

27. Get the first date and last date of the current month.

```
import datetime

def get_first_and_last_date_of_current_month():
    today = datetime.date.today()
    first_day_of_month = today.replace(day=1)
    last_day_of_month = today.replace(day=calendar.monthrange(today.year, today.month)[1])
    return first_day_of_month, last_day_of_month

first_date, last_date = get_first_and_last_date_of_current_month()
print(f"First date: {first_date}")
print(f"Last date: {last_date}")
```

```
First date: 2022-12-01
Last date: 2022-12-31
```

28. Get me the 1st and last date of the current month in the format as following. '14th June 2016 Tuesday 10:00:00 AM'

```
import datetime
```

```
def get_first_and_last_date_of_current_month():
    today = datetime.date.today()
    first_day_of_month = today.replace(day=1)
    last_day_of_month = today.replace(day=calendar.monthrange(today.year, today.month)[1])
    return first_day_of_month, last_day_of_month

first_date, last_date = get_first_and_last_date_of_current_month()
first_date_str = first_date.strftime('%dth %B %Y %A %I:%M:%S %p')
last_date_str = last_date.strftime('%dth %B %Y %A %I:%M:%S %p')

print(f"First date: {first_date_str}")
print(f"Last date: {last_date_str}")
```

```
First date: 01th December 2022 Thursday 12:00:00 AM
Last date: 31th December 2022 Saturday 12:00:00 AM
```

29. Get me random number from 1 to 100.

```
import random

random_number = random.randint(1, 100)
print(random_number)
```

```
23
```

30. Get me a random combination of 4 different numbers between 1 to 100.

```
import random

random_combination = random.sample(range(1, 101), 4)

print(random_combination)
```

```
[51, 27, 33, 6]
```

31.You have a sorted list from 1 to 10 you have to unsort it.

```
import random

sorted_list = list(range(1, 11))
random.shuffle(sorted_list)

print(sorted_list)
```

```
[3, 9, 4, 5, 10, 1, 2, 7, 6, 8]
```

32. Execute a shell script command from python code.

```
import subprocess
subprocess.call(['sh', './test.sh'])
```

```
127
```

33. Create a regular expression to check a valid URL.

```
import re

pattern = r"^(?:http(s)?:\/\/)?[\w.-]+(?:\.[\w\.-]+)+[\w\-\._~:/?#[\]@!\$&'\(\)\*\+,;=.]+$"
string = 'https://colab.research.google.com/drive/1JwoECQw8lD5LcE1VMoaVgH9isWYKOoUO#scrollTo=ZLUGbLyb5PqV'

if re.match(pattern, string):
    print('Valid URL')
else:
    print('Invalid URL')
```

```
Valid URL
```