The Ultimate Python Course - 100 days of Programming (CodeWithHarry)

Playlist Link: [100 days of Code (https://youtube.com/playlist?list=PLu0W_9lII9agwh1XjRt242xIpHhPT2llg)](https://youtube.com/playlist?list=PLu0W_9lII9agwh1XjRt242xIpHhPT2llg)

#3 Modules and pip

2 types of modules - internal (built-in) and external (google it to get the list of modules).

To install an external module: '**pip install pandas**' in the command prompt.

The environment of python in the basic interpreter is an repl (Read Evaluate Print Loop). Here the program is run in shells. But we need to write the programs as a scripts.

In [1]:
```
import pandas
import sklearn
#import tensorflow
```

#4 First Python Program

In [2]:
```
print('Hello World!')
print(5*6-9/2)
```

```
Hello World!
25.5
```

#5 Comments, Escape Sequences and Print Statement

In [3]:
```
print('this is the first line and \n this is the second line')
# the first comment

'''
this is a multiline comment
'''
print('Select and use \t \' ctrl+/ \' \t to comment out multiple lines.')
print('\nEscape sequence - Use \ before any command in print.')

print('\na', 'b', 'c', 'd', 5*2+4-6/2, 3.14, sep=' ~ ', end='\t')
print('a tab insted of a new line for a new print statement. See the previous line.')
print('end=\'\\n\' and sep=\' \' by default')
```

```
this is the first line and 
 this is the second line
Select and use 	 ' ctrl+/ ' 	 to comment out multiple lines.

Escape sequence - Use \ before any command in print.

a ~ b ~ c ~ d ~ 11.0 ~ 3.14	a tab insted of a new line for a new print statement. See the previous line.
end='\n' and sep=' ' by default
```

#6 Variables and Data Types

Variable is like a container which contains data.

```
In [4]: a, b, c, d, e = 12, 3.22, 3+2j, 'skp', True
        print(a+c)
        print('Data Types:', a, type(a), b, type(b), c, type(c), d, type(d), e, type(e))
        list1 = [4,2,6,2.3,c,d,e,[6,2],(2,6)]
        tuple1 = (6.3,8,2,7,d,list1)
        dict1 = {'integer': 32, 'float': 3.6, 'complex':c, 'string':d, 'list': list1, 'tuple': tupl
        print(type(list1), type(tuple1), type(dict1), sep='\t')
        print('All data types in python is OBJECT.')
```

```
(15+2j)
Data Types: 12 <class 'int'> 3.22 <class 'float'> (3+2j) <class 'complex'> skp <class 'st
r'> True <class 'bool'>
<class 'list'>	<class 'tuple'>	<class 'dict'>
All data types in python is OBJECT.
```

#7 Exercise 1: Calculator using Python

```
In [5]: print(45/4, 45//4, 45%4)
```

```
11.25 11 1
```

```
a = eval(input('input number a: '))
print(a)
op = input('Type your desired operation: \n add, sub, mul, div, power, div ans,
remainder')
print(op)
b = eval(input('input number b: '))
print(b)
print(Answer)
if op == 'add':
    print(a+b)
if op == 'sub':
    print(a-b)
if op == 'mul':
    print(a*b)
if op == 'div':
    print(a/b)
if op == 'power':
    print(a**b)
if op == 'div ans':
    print(a//b)
if op == 'remainder':
    print(a%b)
```

#8 Calculator using python (Solution)

In [6]:
```
a, b = 40, 6  # input the values
print('a =', a)
print('b =', b)
print()
print('addition: \t', 'a + b =', a+b)
print('subtraction: \t', 'a - b =', a-b)
print('multiplication:\t', 'a * b =', a*b)
print('division: \t', 'a / b =', a/b)
print('division answer:', 'a // b =', a//b)
print('remainder: \t', 'a %\ b =', a%b)
print('power: \t\t', 'a ** b =', a**b)
```

```
a = 40
b = 6

addition:        a + b = 46
subtraction:     a - b = 34
multiplication:  a * b = 240
division:        a / b = 6.666666666666667
division answer: a // b = 6
remainder:       a %\ b = 4
power:           a ** b = 4096000000
```

#9 Typecasting in Python

Conversion of one data-type from other data-type. Types of type-casting:

1. Explicit conversion (done by user).
2. Implicit conversion (done by python automatically).

Functions: **int(), float(), hex(), oct(), str()**, etc.

In [7]:
```
a, b = '1', '4'
print('explicit typecasting:', a+b, int(a)+int(b))
print('implicit typecasting:', 1.9 + 5)
```

```
explicit typecasting: 14 5
implicit typecasting: 6.9
```

#10 User Input in Python

```
a = input('It\'s string by default')
print(a*3)

b = int(input('input an integer: '))
print(5*b)

c = float(input('input float data: '))
print(c*8)

d = complex(input('input complex number: '))
print(d*2)

e = eval(input('input a number: '))
print(e)
```

#11 Strings

In [8]:
```
a, b, c, d, e = 'Aa1i', 'Bb2ii', 'Cc3iii', 'Dd4iv', 'Ee5v'
print('the numbering is ' + a+b+c)
mult = '''It's a multiline string.
It's the first line.
It's the second line.'''
print(mult)
print('going inside a string:', a[1], c[0], mult[10])
print('String is like an array of characters!')
print('for loop:')
for character in a:
    print(character)
if 'man' in 'Suman':
    print('Yes.')
else:
    print('No.')
```

```
the numbering is Aa1iBb2iiCc3iii
It's a multiline string.
It's the first line.
It's the second line.
going inside a string: a C t
String is like an array of characters!
for loop:
A
a
1
i
Yes.
```

#12 Strings Slicing

string[m:n]: $m, m+1, m+2, \ldots, n-1; \quad (m < n)$

In [9]:
```
order = 'first, second, third, fourth, fifth, sixth, seventh'
orderlen = len(order)
print(order[0:5], order[15:20], orderlen)
print(order[:20], order[:orderlen])
print(order[0:-8], ',\n by interpreter:', order[0:len(order)-8])
print(order[-21:-8], ',\n by interpreter:',
        order[len(order)-21:len(order)-8])
```

```
first third 51
first, second, third first, second, third, fourth, fifth, sixth, seventh
first, second, third, fourth, fifth, sixth, ,
 by interpreter: first, second, third, fourth, fifth, sixth,
fifth, sixth, ,
 by interpreter: fifth, sixth,
```

#13 String Methods

Strings are immutable.

Here's Harry's studio setup using Macbook and iphone!

In [10]:
```
order1 = order.upper()
print('upper:' , order1, ', \nlower:', order1.lower())

s1 = '!!!string 1!!!!'
print(s1, ', rstrip:', s1.rstrip('!'), ', replace:', s1.replace('srting', 'str'))
# replace is not working
print('str to list by split:', order.split(', '))
blogHeading = "introduction to jS"
print('capitalize:', blogHeading.capitalize())
s1c = s1.center(25)
print('center:', s1, s1c, len(s1), len(s1c))
print('a.center(n) adds n-len(a) extra spaces and aligns the string at the center.')

print('count:', s1.count('!!'), s1c.count(' '))
print('endswith:' , order.endswith('sixth', 10, 42))
print('startswith:', order.startswith('first'))
print('find:', order.find('fifth'), order[30], order.find('fh'))
print('find:', order.index('fifth'), order[30])

print('isalnum (alphabetic and numeric only):', a.isalnum(), order.isalnum())
print('isalpha (alphabetic only):', a.isalpha(), 'welcome'.isalpha())
print('islower (all lower):', order.islower(), a.islower())
print('isupper (all upper):', order1.isupper(), a.isupper())
print('isprintable (no \n like characters):', order.isprintable(), mult.isprintable())
print('isspace (only spaces present):', order.isspace(), '  '.isspace())
print('istitle (starting of every word is capital or not):',
        'World Health Organization'.istitle(), 'Intro to Python'.istitle())
print('title (converts a string into title):', 'python basics'.title())
print('swapcase (upper to lower and vice-versa):', a.swapcase())
```

```
upper: FIRST, SECOND, THIRD, FOURTH, FIFTH, SIXTH, SEVENTH ,
lower: first, second, third, fourth, fifth, sixth, seventh
!!!string 1!!!! , rstrip: !!!string 1 , replace: !!!string 1!!!!
str to list by split: ['first', 'second', 'third', 'fourth', 'fifth', 'sixth', 'seventh']
capitalize: Introduction to js
center: !!!string 1!!!!      !!!string 1!!!!      15 25
a.center(n) adds n-len(a) extra spaces and aligns the string at the center.
count: 3 11
endswith: True
startswith: True
find: 30 f -1
find: 30 f
isalnum (alphabetic and numeric only): True False
isalpha (alphabetic only): False True
islower (all lower): True False
isupper (all upper): True False
isprintable (no
 like characters): True False
isspace (only spaces present): False True
istitle (starting of every word is capital or not): True False
title (converts a string into title): Python Basics
swapcase (upper to lower and vice-versa): aA1I
```

#14 if-else Conditionals

Conditions: <, >, <=, >=, ==. After condition we use an indentation (tab) in the next line.

```
age = int(input('input your age: '))
print('Your age:', age)
if age>18:
    print('You can drive.')
elif age==18:
    print('Make your driving lisence.')
else:
    print('You can\'t drive.')
```

```
data  = eval(input('Input the value to check the type: '))
print('The value is', data)
```

```
if type(data)==int:
    print('It\'s an integer.')
elif type(data)==float:
    print('It\'s a float.')
elif type(data)==complex:
    print('It\'s a complex number.')
elif type(data)==str:
    print('It\'s a string.')
elif type(data)==list:
    print('It\'s a list.')
elif type(data)==tuple:
    print('It\'s a tuple.')
elif type(data)==dict:
    print('It\'s a dictionary.')
elif type(data)==bool:
    print('It\'s a boolean logic.')
else:
    print('I don\'t know the data type.')
```

```
sleephours = eval(input('How many hours do you sleep in a day?'))
print(sleephours, 'hours of sleep in a day.')
if sleephours<=3:
    print('Please take rest!')
elif 3<sleephours<=9:
    print('It\'s good.')
    if 3<sleephours<6.5:
        print('Try to increase it by 1 hour.')
    elif 6.5<=sleephours<8:
        print('This duration is perfect!')
    else:
        print('Try to decrease it by 1 hour.')
elif sleephours>9:
    print('Aur kitna soyega!')
else:
    print('Please give your input again.')
```

#15 Exercise 2: Good Morning

In [11]:
```
import time

timestamp = time.strftime('%H:%M:%S.')
print('The time is', timestamp)
timestamph= int(time.strftime('%H'))
timestampm = int(time.strftime('%M'))
print('hour:', timestamph, ', minutes:', timestampm)
#timestamph = 0

if 3<timestamph<12:
    print('Good Morning!')
elif 15<=timestamph<18:
    print('Good Afternoon!')
elif 18<=timestamph<=21:
    print('Good Evening!')
elif 21<timestamph<23 or 0<=timestamph<=3:
    print('Good Night!')
else:
    print('Have a nice day!')
```

```
The time is 09:14:20.
hour: 9 , minutes: 14
Good Morning!
```

#16 Match Case Statements in Python

Requires **Python 3.10** version.

```
# see the .py file.
x = int(input('enter an integer: '))

match x:
    case 0:
        print('x is 0.')
    case 4:
        print('x is 4.')
    case _:
        print(x)
```

#17 for Loops

In [12]:
```
s2 = 'SKP'
for s in s2:
    print(s)
    if s=='S':
        print('Suman')
    if s=='P':
        print('Pal')
```

```
S
Suman
K
P
Pal
```

In [13]:
```python
list2 = ['list', 61,3.569, 7+7j]
for l in list2:
    print(l)
print()
white = ['Violet', 'Indigo', 'Blue', 'Green', 'Yellow', 'Orange', 'Red']
for color in white:
    print(color)
    for i in color:
        print(i)
```

```
list
61
3.569
(7+7j)

Violet
V
i
o
l
e
t
Indigo
I
n
d
i
g
o
Blue
B
l
u
e
Green
G
r
e
e
n
Yellow
Y
e
l
l
o
w
Orange
O
r
a
n
g
e
Red
R
e
d
```

In [14]:
```python
tuple2 = ('tuple',2,6.8,5-2j)
for t in tuple2:
    print(t)
```

```
tuple
2
6.8
(5-2j)
```

In [15]:
```
dict2 = {'a':56, 'b':14, 'c':75}
for d in dict2:
    print(d)
```

```
a
b
c
```

In [16]:
```
for i in range(1, 6):
    print(i**2)
for k in range(5):
    print(k+1)
for i in range(12,40,4):
    print(i)
```

```
1
4
9
16
25
1
2
3
4
5
12
16
20
24
28
32
36
```

#18 while Loops

9:20: while loop to keep ants away!!!

do while: The loop runs 1st time automatically. If the condition is the true, the loop continues running.

In [17]:
```
i = 0
while (i<5): # checks the condition after each iteration
    print(i+1)
    i = i+2
```

```
1
3
5
```

```
i = int(input('guess an integer greater than given limit:  '))
while i<=120:
    i = int(input('guess an integer greater than given limit:  '))
    print('Your Guess:', i)

print('Done with the loop. The upper limit was 120.')
```

In [18]:
```
count = 5 # input
while (count>0):
    print(count)
    count = count -1
else:
    print('I am inside else.')
```

```
5
4
3
2
1
I am inside else.
```

In [19]:
```
# do while loop

i = 0
while True:
    print(i)
    i = i+1
    if i%10==0:
        break
```

```
0
1
2
3
4
5
6
7
8
9
```

#19 break and continue

break: Leave the loop.

continue: Leave the iteration.

In [20]:
```
for i in range(20):
    if i==5:
        print('skip the iteration')
        continue
    print('2 ^',i,'=', 2**i)
    if i==10:
        print('skip the loop')
        break
```

```
2 ^ 0 = 1
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
skip the iteration
2 ^ 6 = 64
2 ^ 7 = 128
2 ^ 8 = 256
2 ^ 9 = 512
2 ^ 10 = 1024
skip the loop
```

#20 Functions

2 types:

1. built-in functions: print(), min(), max(), sum(), type(), range(), list(), tuple(), dict(), set() etc.
2. user defined function.

In [21]:
```
a, b = 5, 2
gmean2 = (a*b)/(a+b)
print(gmean2)

def gmean2f(a, b):
    print('a =', a, '\t b =', b)
    if a>b:
        print('first number is greater')
    elif a<b:
        print('second number is greater')
    else:
        print('the numbers are equal')
    gmean = (a*b)/(a+b)
    print('geometric mean =', gmean)

gmean2f(3,6)

def fn1(x, y):
    pass
print(fn1(1,3))
```

```
1.4285714285714286
a = 3    b = 6
second number is greater
geometric mean = 2.0
None
```

#21 Function Arguments

In [22]:
```
def average2(a=5, b=10): # Default values are given. We can change it.
    print('a =', a, '\t b =', b)
    print('The average is', (a+b)/2)

average2() # default arguments
average2(56)
average2(b=16)
average2(23,65)
average2(b=12, a=45)  # keyword arguments
# no required arguments
```

```
a = 5    b = 10
The average is 7.5
a = 56   b = 10
The average is 33.0
a = 5    b = 16
The average is 10.5
a = 23   b = 65
The average is 44.0
a = 45   b = 12
The average is 28.5
```

In [23]:
```
def average(*numbers):
    print(type(numbers))
    s = 0
    for i in numbers:
        s = s + i
    return s/len(numbers) # return to calling function
    return 0 # no use

print(average(4,7,2,6,3,5))

def name(**name):
    print(type(name))
    print('Hello,', name['first'], name['middle'], name['last'])

name(first='Suman', middle='Kumar', last='Pal')
```

```
<class 'tuple'>
4.5
<class 'dict'>
Hello, Suman Kumar Pal
```

#22 Lists

Slicing default: **a[0:len(a):1]**.

In [24]:
```
l1 = [5,3,7,9,2,6, 'skp',0]
print(l1, type(l1), len(l1))
print(l1[3])
print('negative indexing:', l1[-4], l1[len(l1)-4])

if 0 in l1:
    print('0 is present.')
else:
    print('0 is not present.')

print('slicing:', l1[:100], l1[3:7], l1[:-2], l1[-4:-2], l1[::2])
print('list comprehension:')
l0 = []
print(l0)
l2 = [i**2+1 for i in range(4)]
print(l2)
l3 = [i**2 for i in range(10) if i%2==0]
print(l3)
```

```
[5, 3, 7, 9, 2, 6, 'skp', 0] <class 'list'> 8
9
negative indexing: 2 2
0 is present.
slicing: [5, 3, 7, 9, 2, 6, 'skp', 0] [9, 2, 6, 'skp'] [5, 3, 7, 9, 2, 6] [2, 6] [5, 7, 2,
'skp']
list comprehension:
[]
[1, 2, 5, 10]
[0, 4, 16, 36, 64]
```

#23 List Methods

```
In [25]: l1 = [7,2,9,3,9,0,0,6,0,'str1',8,'str2',7]
         l1.append('new')
         print('append:', l1)

         l4 = [5,8,68,3,8,0,3,85,36]
         print(l4)
         l4.sort()
         print('sort:', l4)
         l3.sort(reverse=True)
         print(l3)
         l1.reverse()
         print('reverse:', l1)
         print('index:', l1.index(0)) # index of 0
         print('count:', l1.count(0))
         l2 = [3,5,2,6,9,0]
         l2p = l2  # view
         l2c = l2.copy() # copy
         l2[0] = 'changed'
         print('view and copy:', l2, l2p, l2c)
         l1.insert(5, 'inserted in index 5')
         print('insert:', l1)
         l2ex = [100,200,300,400]
         l2.extend(l2ex)
         print('extend:', l2)
         l23 = l2 + l3
         print('addition:', l23)
```

```
append: [7, 2, 9, 3, 9, 0, 0, 6, 0, 'str1', 8, 'str2', 7, 'new']
[5, 8, 68, 3, 8, 0, 3, 85, 36]
sort: [0, 3, 3, 5, 8, 8, 36, 68, 85]
[64, 36, 16, 4, 0]
reverse: ['new', 7, 'str2', 8, 'str1', 0, 6, 0, 0, 9, 3, 9, 2, 7]
index: 5
count: 3
view and copy: ['changed', 5, 2, 6, 9, 0] ['changed', 5, 2, 6, 9, 0] [3, 5, 2, 6, 9, 0]
insert: ['new', 7, 'str2', 8, 'str1', 'inserted in index 5', 0, 6, 0, 0, 9, 3, 9, 2, 7]
extend: ['changed', 5, 2, 6, 9, 0, 100, 200, 300, 400]
addition: ['changed', 5, 2, 6, 9, 0, 100, 200, 300, 400, 64, 36, 16, 4, 0]
```

#24 Tuples

It is immutable.

```
In [26]: t1 = (4,)
         t0 = ()
         print(t1, type(t1), t0, type(t0))
         t2 = (5,3,6,2,'str1',4-2j)
         print(t2, t2[3], t2[-4], t2[len(t2)-4])
         if ('str1' in t2):
             print('Yes.')
         else:
             print('No.')
         t3 = (5,8,3,0,8,0,6,8,2,7,9,1)
         print(t3)
         print(t3[3:8], t3[1:9:2])
         t2c = t2[1:4]
         print('copy:', t2, t2c)
```

```
(4,) <class 'tuple'> () <class 'tuple'>
(5, 3, 6, 2, 'str1', (4-2j)) 2 6 6
Yes.
(5, 8, 3, 0, 8, 0, 6, 8, 2, 7, 9, 1)
(0, 8, 0, 6, 8) (8, 0, 0, 8)
copy: (5, 3, 6, 2, 'str1', (4-2j)) (3, 6, 2)
```

#25 Tuple Methods

Tuples can't be changed directly. We can manipulate it by changing it into a list.

In [27]:
```
countries = ('USA', 'Russia', 'India', 'England', 'China')
temp = list(countries) # converting into a list
print(countries, temp, sep='\n')
temp.append('Germany') # adding a str at the end of the list
temp.pop(3)  # removing str in temp[3]
temp[3] = 'France' # changing a str
countries = tuple(temp)
print(temp, countries, sep='\n')

countriesp = ('Israel', 'Japan', 'Spain')
print(countriesp)
print('merge:', countries + countriesp)

res = t3.index(0)
res = t3.count(0)
print('count (of an element in a tuple):', res)
print(t3, len(t3))
res1 = t3.index(0, 4, 10) # finding in a range by slicing
print('index:', res, res1)
```

```
('USA', 'Russia', 'India', 'England', 'China')
['USA', 'Russia', 'India', 'England', 'China']
['USA', 'Russia', 'India', 'France', 'Germany']
('USA', 'Russia', 'India', 'France', 'Germany')
('Israel', 'Japan', 'Spain')
merge: ('USA', 'Russia', 'India', 'France', 'Germany', 'Israel', 'Japan', 'Spain')
count (of an element in a tuple): 2
(5, 8, 3, 0, 8, 0, 6, 8, 2, 7, 9, 1) 12
index: 2 5
```

#26 Exercise 2: Solution

In [28]:
```
import time
t = time.strftime('%H:%M:%S')
print(t, type(t))
hour = time.strftime('%H')
hr = int(hour)
print(hour, type(hour), hr, type(hr))

if (hr>0 and hr<12):
    print('Good Morning!')
elif (hr>12 and hr<17):
    print('Good Afternoon!')
elif (hr>17 and hr<0):
    print('Good Night!')
```

```
09:14:21 <class 'str'>
09 <class 'str'> 9 <class 'int'>
Good Morning!
```

#27 Exercise 3: Kaun Banega Crorepati (KBC)

Create a program capable of displaying questions to the user like KBC.

1. Use list data type to store and their correct answers.
2. Display the final amount the person is taking home after playing the game.

```
qs1 = ['What do we use in python to have a better controls on data than excel?',
        'numpy', 'numba', 'pandas', 'seaborn']
ans1, corropt1 = qs1[3], 3
```

```
qs2 = ['Which branch of physics is related to god particles?',
       'astrophysics', 'condensed matter physics', 'geophysics', 'higher energy
physics']
ans2, corropt2 = qs2[4], 4
qs3 = ['Which of these is the fastest?',
       'c-python', 'jython', 'pypy', 'java']
ans3, corropt3 = qs3[3], 3
qs4 = ['Which is considered to be a middle level language?',
       'c', 'c++', 'java', 'assembly langauge']
ans4, corropt4 = qs4[1], 1
qs5 = ['Konsi saal Ohio ne \'Shreya Ghoshal Day\' announce kiya hai?',
       '2009', '2010', '2011', '2012']
ans5, corropt5 = qs5[2], 2
qs6 = ['Inme se kisko marne ke baad bharatratna diya gaya hai?',
       'C. V. Raman', 'Rajiv Gandhi', 'Indira Gandhi', 'Laal Bahadur Shastri']
ans6, corropt6 = qs6[2], 2
qs7 = ['Who first used matrix formulations in quantum mechanics?',
       'Werner Heisenberg', 'Erwin Schrodinger', 'Paul Dirac', 'Louis de Broglie']
ans7, corropt7 = qs7[1], 1
qs8 = ['Who ranked 3rd among Times top persons in 20th century?',
       'A. Hitler', 'A. Einstein', 'Roosevelt', 'M. K. Gandhi']
ans8, corropt8 = qs8[4], 4
qs9 = ['In Bollywood, who holds the record for recording maximum songs in a day?',
       'Kumar Sanu', 'Udit Narayan', 'Arijit Singh', 'Kishore Kumar']
ans9, corropt9 = qs9[1], 1
qs10 = ['What was Einstein\'s age at the time of his death?',
        '73', '76', '79', '82']
ans10, corropt10 = qs10[2], 2

qs = [qs1, qs2, qs3, qs4, qs5, qs6, qs7, qs8, qs9, qs10]
ans = [ans1, ans2, ans3, ans4, ans5, ans6, ans7, ans8, ans9, ans10]
corropt = [corropt1, corropt2, corropt3, corropt4, corropt5, corropt6, corropt7,
           corropt8, corropt9, corropt10]

print('''Devioyon aur sajjano, Kaun banega crorepati mein aap sab ka swagat hai.
Soch samajh ke jawab dijiyega. Sahi jawab ke liye apko paisa milega
aur galat jawab ke liye apke balance mein se paisa ghatega.
Welcome to KBC.''')

balance = 0
starting_reward = int(input('Please select starting reward. 500/5000/50000: '))

for i in range(len(ans)):
    rs1 = starting_reward*(i+1)
    print('\nsawal', rs1,'rs ke liye:\n', qs[i][0])
    for opt in range(1,5):
        print(opt, qs[i][opt])
    a = int(input('enter 1, 2, 3, 4: '))
    if a==corropt[i]:
        print(ans[i], 'is the correct answer. Aur app jite hai', rs1, 'rs.')
        balance += rs1
    else:
        print(ans[i], 'is the correct answer.')
        balance += -rs1/2
    print('\tbalance =', balance, 'rs.')

print('\nTotal winning amount:', balance, 'rs.',
      '\nItna paisa nahi hai bhai. Bachchan Saab ke paas jao. Wo de denge.')
```

#28 f-strings

The operations done by *format* can be done more easily using **f-strings**. It was introduced in python 3.6.

In [29]:
```
str1 = 'Hey my name is {1} and I am from {0}.'
print(str1)
myname = 'Suman Kumar Pal'
address = 'West Bengal, India'
print(str1.format(address, myname))
print(f'Hey my name is {myname} and I am from {address}')
pival = 3.141592653589793
print('pi =', pival)
pivalstr = 'Value of pi upto 4 decimals is {pi_value: 0.4f}.'
print(pivalstr.format(pi_value=pival))
print(f'Value of pi upto 7 decimals is {pival:0.7f}.')
print(type(f'We can do calculations inside strings by using this. Ex. {2*63-45*3}.'))
print(f'We use f-strings like this: Value of pi upto 4 decimals is {{pi_value: 0.4f}}.')
```

```
Hey my name is {1} and I am from {0}.
Hey my name is Suman Kumar Pal and I am from West Bengal, India.
Hey my name is Suman Kumar Pal and I am from West Bengal, India
pi = 3.141592653589793
Value of pi upto 4 decimals is  3.1416.
Value of pi upto 7 decimals is 3.1415927.
<class 'str'>
We use f-strings like this: Value of pi upto 4 decimals is {pi_value: 0.4f}.
```

#29 DocStrings

Python ignores the comments but it gives a special treatment to docstrings. It can be accessed by **__doc__** attribute. Docstrings can also effect the output sometime. We can give description of a function using docstrings just before or after the first line (i.e. `def fuction():` line).

PEP 8: It's a document that proposes guidelines and best practices on how to write python code. The full form is Python Enhancement Proposal.

Run in command prompt:

1. python
2. import this (Get *The Zen of Python* poem)

In [30]:
```
def square(n):
    'Takes a number n and returns the square of n' # docstring
    print(n**2)
square(25)
print(square.__doc__)
```

```
625
Takes a number n and returns the square of n
```

#30 Recursion

It's a process of defining something in terms of itself.

In [31]:
```python
# code for factorial
# factorial(n) = n* factorial(n-1)
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n* factorial(n-1)
print(factorial(7))

# quick quiz
print('Fibonacci number:')
def Fibonacci_num(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return Fibonacci_num(n-1)+Fibonacci_num(n-2)
for i in range(10):
    print('n =', i, ', Fibonacci_num =', Fibonacci_num(i))
```

```
5040
Fibonacci number:
n = 0 , Fibonacci_num = 0
n = 1 , Fibonacci_num = 1
n = 2 , Fibonacci_num = 1
n = 3 , Fibonacci_num = 2
n = 4 , Fibonacci_num = 3
n = 5 , Fibonacci_num = 5
n = 6 , Fibonacci_num = 8
n = 7 , Fibonacci_num = 13
n = 8 , Fibonacci_num = 21
n = 9 , Fibonacci_num = 34
```

#31 Sets

Set is a (unordered) collection of well defined objects. Sets are unchangeable. Sets don't contain duplicate items. As the order is not maintained in sets, we can use indexing and slicing.

In [32]:
```python
set1 = {3,6,2,9,3,0,1+6,5+6j,'skp',True}
print(set1, type(set1))
emp1 = {}
emp2 = set()
print(type(emp1), type(emp2))
for value in set1:
    print(value)
```

```
{0, True, 2, 3, 6, 7, 9, 'skp', (5+6j)} <class 'set'>
<class 'dict'> <class 'set'>
0
True
2
3
6
7
9
skp
(5+6j)
```

#32 Set Methods

1. symmetric difference: $A \cup B - A \cap B$.
2. difference: $A - B$.
3. disjoint: $A - B = 0$.
4. superset: A contains B.
5. subset: A is contained in B.

6. remove vs discard: If the item we want to delete is not present in the set, remove shows error but discard shows no errors here.
7. pop: Deletes a random value. We can access that value.

In [33]:
```
set1 = {2,0,5,0,6,'skp', 1+3j}
set2 = {2,8,3,9,7,0,12,'skp'}
set3 = {0,1,2,3,7}
print(set1, set2, set3)
print('union:', set1.union(set2))
print(set1)
set1.update(set2)
print('update:', set1)
print('intersection:', set1.intersection(set2))
print(set1)
set1.intersection_update(set2)
print('intersection_update:', set1)

set2sd3 = set2.symmetric_difference(set3)
print('\nsymmetric_difference:', set2sd3)
set1.symmetric_difference_update(set2)
print('symmetric_difference_update:', set1)
set2d3 = set2.difference(set3)
print('difference:', set2d3)
set2.difference_update(set3)
print('difference_update:', set2)

print('\nisdisjoint:', set2.isdisjoint(set3))
print('issuperset:', set2.issuperset(set1))
print('issubset:', set1.issubset(set3))

print(set1)
set1.add(2)
print('add:', set1)
print(set3)
set3.remove(0)
print('remove:', set3)
print(set2)
set2.discard(0)
print('discard:', set2)
item = set2.pop()
print('pop:', set2, item)
del set1
print('set1 is deleted.')
set2.clear()
print('clear:', set2)
if 0 in set3:
    print('set3 contains 0.')
else:
    print('set3 don\'t contain 0.')
```

```
{0, 2, 5, 6, 'skp', (1+3j)} {0, 2, 3, 7, 8, 9, 12, 'skp'} {0, 1, 2, 3, 7}
union: {0, 2, 3, 5, 6, 7, 8, 9, (1+3j), 12, 'skp'}
{0, 2, 5, 6, 'skp', (1+3j)}
update: {0, 2, 3, 5, 6, 7, 8, 9, (1+3j), 12, 'skp'}
intersection: {0, 2, 3, 7, 8, 9, 12, 'skp'}
{0, 2, 3, 5, 6, 7, 8, 9, (1+3j), 12, 'skp'}
intersection_update: {0, 2, 3, 7, 8, 9, 12, 'skp'}

symmetric_difference: {1, 8, 9, 12, 'skp'}
symmetric_difference_update: set()
difference: {8, 9, 12, 'skp'}
difference_update: {8, 9, 12, 'skp'}

isdisjoint: True
issuperset: True
issubset: True
set()
add: {2}
{0, 1, 2, 3, 7}
remove: {1, 2, 3, 7}
{8, 9, 12, 'skp'}
discard: {8, 9, 12, 'skp'}
pop: {9, 12, 'skp'} 8
set1 is deleted.
clear: set()
set3 don't contain 0.
```

#33 dictionaries

Dictionaries are ordered from python 3.7 onwards.

In [34]:
```
print(dict1['string'])
dict2 = {'c': 'speed of light', 'G': 'gravitational constant',
        'h': 'Planck\'s constant'}
print(dict2)
print(dict2['h'])
#print(dict2['a']) # KeyError
print(dict2.get('a'))
print('keys:', dict2.keys(), type(dict2.keys()))
print('values:', dict2.values(), type(dict2.values()))
for key in dict2.keys():
    print(dict2[key])
for key in dict2.keys():
    print(f'The value corresponding to the key {key} is {dict2[key]}.')
print('items:', dict2.items(), type(dict2.items()))
for key, val in dict2.items():
    print(key, val)
```

```
skp
{'c': 'speed of light', 'G': 'gravitational constant', 'h': "Planck's constant"}
Planck's constant
None
keys: dict_keys(['c', 'G', 'h']) <class 'dict_keys'>
values: dict_values(['speed of light', 'gravitational constant', "Planck's constant"]) <cl
ass 'dict_values'>
speed of light
gravitational constant
Planck's constant
The value corresponding to the key c is speed of light.
The value corresponding to the key G is gravitational constant.
The value corresponding to the key h is Planck's constant.
items: dict_items([('c', 'speed of light'), ('G', 'gravitational constant'), ('h', "Planc
k's constant")]) <class 'dict_items'>
c speed of light
G gravitational constant
h Planck's constant
```

#34 dictionary methods

```
In [35]: dict2a = {'hcut': 'reduced Planck\'s constant',
                 'e0': 'absolute permittivity', 'mu0': 'absolute permeability',
                 'R': 'universal gas constant'}
         dict2.update(dict2a)
         print(dict2.keys())
         dict2a.clear()
         print('clear:', dict2a)
         dict2.pop('hcut')
         print('pop:', dict2.keys())
         dict2.popitem() # pops the last item
         print('popitem:', dict2.keys())
         del dict2a
         print('dict2a is deleted.')
         del dict2['e0']
         del dict2['mu0']
         print(dict2.keys())
         print('Google python dictionary documentation.')
```

```
dict_keys(['c', 'G', 'h', 'hcut', 'e0', 'mu0', 'R'])
clear: {}
pop: dict_keys(['c', 'G', 'h', 'e0', 'mu0', 'R'])
popitem: dict_keys(['c', 'G', 'h', 'e0', 'mu0'])
dict2a is deleted.
dict_keys(['c', 'G', 'h'])
Google python dictionary documentation.
```

#35 for loop with else

`else` loop can also be used along with `for` loops and `while` loops. If loop ends successfully, `else` is executed. But if we have `break` in the loop, `else` is not executed.

In [36]:
```python
for elt in range(0):
    print(elt)
else:
    print('No more elt is present.')

for i in range(100):
    print(i)
    if i==4:
        break
else:
    print('else is executing.')

print('')
i = 2
while i<7:
    print(i)
    i = i+1
else:
    print('else is executing')
i = 2
while i<7:
    print(i)
    i = i+1
    if i==5:
        break
else:
    print('else is executing')

for x in range(5):
    print('iteration no. {} in for loop'.format(x+1))
else:
    print('else block in loop')
print('out of loop')
for x in range(5):
    print('iteration no. {} in for loop'.format(x+1))
    if x==3:
        break
else:
    print('else block in loop')
print('out of loop')
```

```
No more elt is present.
0
1
2
3
4

2
3
4
5
6
else is executing
2
3
4
iteration no. 1 in for loop
iteration no. 2 in for loop
iteration no. 3 in for loop
iteration no. 4 in for loop
iteration no. 5 in for loop
else block in loop
out of loop
iteration no. 1 in for loop
iteration no. 2 in for loop
iteration no. 3 in for loop
iteration no. 4 in for loop
out of loop
```

#36 Exception Handling

try, except is used here.

```
try:
    a = int(input('enter a number to get multiplication table: '))
    print(f'Multiplication table of {a} is:')

    for i in range(1,11):
        print(f'{a} X {i} = {a*i}')
except Exception as e:
     print('Error in the code:', e)
# except:
#     print('Error is occured here!')

try:
    a = int('j')
except IndexError:
    print('index error here.')
except ValueError:
    print('value error here.')

print('Some lines of code.')
print('End of program.')
```

#37 finally Keyword

In [37]:
```
try:
    print(l1)
    print(l1[15])
except Exception as e:
    print('Error!:', e)
finally:
    print('It is always executed.')
print('It is also executed here.')
print('')

def func1():
    try:
        print(l1)
        print(l1[15])
        return 1
    except:
        print('Some error ocurred.')
        return 0
    finally:
        print('It is always executed (even after return!).') # executed
    print('Check if it\'s executed or not') # not executed
print(func1())
```

```
['new', 7, 'str2', 8, 'str1', 'inserted in index 5', 0, 6, 0, 0, 9, 3, 9, 2, 7]
Error!: list index out of range
It is always executed.
It is also executed here.

['new', 7, 'str2', 8, 'str1', 'inserted in index 5', 0, 6, 0, 0, 9, 3, 9, 2, 7]
Some error ocurred.
It is always executed (even after return!).
0
```

#38 Raising custom errors (raise)

In [38]:
```python
a = int('8')
if a<5 or a>9:
    raise ValueError('Value should be between 5 to 10.')

str2 = 'abc'
if str2 =='stop':
    raise ValueError('Program is stopped.')
```

#39 Exercise 3 solution

```python
qs = [['What do we use in python to have a better controls on data than excel?',
  'numpy',
  'numba',
  'pandas',
  'seaborn', 3],
 ['Which branch of physics is related to god particles?',
  'astrophysics',
  'condensed matter physics',
  'geophysics',
  'higher energy physics', 4],
 ['Which of these is the fastest?', 'c-python', 'jython', 'pypy', 'java', 3],
 ['Which is considered to be a middle level language?',
  'c',
  'c++',
  'java',
  'assembly langauge', 1],
 ["Konsi saal Ohio ne 'Shreya Ghoshal Day' announce kiya hai?",
  '2009',
  '2010',
  '2011',
  '2012', 2],
 ['Inme se kisko marne ke baad bharatratna diya gaya hai?',
  'C. V. Raman',
  'Rajiv Gandhi',
  'Indira Gandhi',
  'Laal Bahadur Shastri', 2],
 ['Who first used matrix formulations in quantum mechanics?',
  'Werner Heisenberg',
  'Erwin Schrodinger',
  'Paul Dirac',
  'Louis de Broglie', 1],
 ['Who ranked 3rd among Times top persons in 20th century?',
  'A. Hitler',
  'A. Einstein',
  'Roosevelt',
  'M. K. Gandhi', 4],
 ['In Bollywood, who holds the record for recording maximum songs in a day?',
  'Kumar Sanu',
  'Udit Narayan',
  'Arijit Singh',
  'Kishore Kumar', 1],
 ["What was Einstein's age at the time of his death?", '73', '76', '79', '82', 2]]

levels = [1000, 2000, 3000, 5000, 10000, 20000, 40000, 80000, 160000, 320000]
money = 0
i = 0
for i in range(0, len(qs)):
    q = qs[i]
    print(f'\nQuestion for Rs. {levels[i]}')
    print(f'Question {i+1}: {q[0]}')
    print(f'a. {q[1]} \t b. {q[2]}')
    print(f'c. {q[3]} \t d. {q[4]}')
    reply = int(input('enter your answer (1/2/3/4) or 0 to quit: '))
    if reply == 0:
        money = levels[i-1]
        break
    if reply == q[-1]:
        print(f'Correct answer, you have won Rs. {levels[i]}')
        if i==4:
            money = 10000
        elif i == 9:
            money = 320000
    else:
        print('Wrong answer')
        break
print(f'Your take home money is {money}.')
```

#40 Exercise 4: Secret Code Language

Write a program to translate a message into secret code language. Use the rules below to translate normal english into secret code language:

Coding:

1. if the word contains atleast 3 characters, remove the first letter and append it at the end. Now append 3 random characters at the starting and the end.
2. else: simply reverse the string.

Decoding:

1. if the word contains less than 3 characters, reverse it.
2. else: remove 3 random characters from start and end. Now remove the last letter and append it to the beginning.

In [ ]:

```

```

#41 Short hand if else statements

It's not useful for long codes. Use it only for short codes.

In [39]:

```
a = 330
b = 33
print('A') if a>b else print('=') if a==b else print('B')
if a>b:
    print('A')
elif a==b:
    print('=')
else:
    print('B')

c = 9 if a>b else 0
print(c)
```

```
A
A
9
```

#42 enumerate

Google *linter*. We can see elements in a list, string, tuple etc. with the corresponding index using `enumerate` .

```
In [40]: list3 = [24,54,62,24,76,84,25,56,73]
         index = 0
         print(list3)
         for l in list3:
             print(l)
             if index ==3:
                 print(f'The value here is {list3[3]}.')
             index += 1

         print('\nenumerate(list3, 11):')

         for index, mark in enumerate(list3, 11):
             print(f'{index}: {mark}')
```

```
[24, 54, 62, 24, 76, 84, 25, 56, 73]
24
54
62
24
The value here is 24.
76
84
25
56
73

enumerate(list3, 11):
11: 24
12: 54
13: 62
14: 24
15: 76
16: 84
17: 25
18: 56
19: 73
```

#43 Virtual Environment

```
In [41]: ## watch it later.
```

#44 import

To import a defined function from a file, the file name should not contain spaces and the file should be a .py file (not .ipynb file). The imported file should be in same folder.

In [42]:
```
import math
from math import sqrt, pi
#from math import * # imports all (not recommended)
import math as m
from math import sqrt as rt

print(dir(math))
#print(dir(m))

print(math.sqrt(25)*pi)
print(m.sin(m.pi/2))
print(rt(81))

print(m.nan, type(m.nan))
```

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',
'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees',
'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp',
'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt',
'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'pe
rm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 't
au', 'trunc', 'ulp']
15.707963267948966
1.0
9.0
nan <class 'float'>
```

In [43]:
```
#import tutorial_day44 as day44
from tutorial_day44 import day44f, str1
#from tutorial_day44 import *
day44f()
print(str1)
```

```
Things related to importing a python is covered here.
1st string in this file
```

#45 if __name__ == "__main__"

The file is not executed for the first time here if this is used. It's important because if the file imported has some functionality to delete a file of the device and `if __name__=='__main__'` is not used at that file, by just importing that python file, the other files of that device can be deleted!

In [44]:
```
print(__name__)
import tutorial_day45
```

```
__main__
```

#46 os Module

See os module documentation. The full form is Operating System. See os.system.

In [45]:
```
import os

if not os.path.exists("100 days python (os)"):
    os.mkdir('100 days python (os)')

for i in range(0, 100):
    if not os.path.exists(f'100 days python (os)/Day {i+1} of 100'):
        os.mkdir(f'100 days python (os)/Day {i+1}')
        os.rename(f'100 days python (os)/Day {i+1}',
                f'100 days python (os)/Day {i+1} of 100')
```

In [46]:
```python
folders = os.listdir("100 days python (os)")
print(folders)

for folder in folders:
    print(folder)
    print(os.listdir(f'100 days python (os)/{folder}'))
```

```
['Day 1 of 100', 'Day 10 of 100', 'Day 100 of 100', 'Day 11 of 100', 'Day 12 of 100',
'Day 13 of 100', 'Day 14 of 100', 'Day 15 of 100', 'Day 16 of 100', 'Day 17 of 100',
'Day 18 of 100', 'Day 19 of 100', 'Day 2 of 100', 'Day 20 of 100', 'Day 21 of 100', 'D
ay 22 of 100', 'Day 23 of 100', 'Day 24 of 100', 'Day 25 of 100', 'Day 26 of 100', 'Da
y 27 of 100', 'Day 28 of 100', 'Day 29 of 100', 'Day 3 of 100', 'Day 30 of 100', 'Day
31 of 100', 'Day 32 of 100', 'Day 33 of 100', 'Day 34 of 100', 'Day 35 of 100', 'Day 3
6 of 100', 'Day 37 of 100', 'Day 38 of 100', 'Day 39 of 100', 'Day 4 of 100', 'Day 40
of 100', 'Day 41 of 100', 'Day 42 of 100', 'Day 43 of 100', 'Day 44 of 100', 'Day 45 o
f 100', 'Day 46 of 100', 'Day 47 of 100', 'Day 48 of 100', 'Day 49 of 100', 'Day 5 of
100', 'Day 50 of 100', 'Day 51 of 100', 'Day 52 of 100', 'Day 53 of 100', 'Day 54 of 1
00', 'Day 55 of 100', 'Day 56 of 100', 'Day 57 of 100', 'Day 58 of 100', 'Day 59 of 10
0', 'Day 6 of 100', 'Day 60 of 100', 'Day 61 of 100', 'Day 62 of 100', 'Day 63 of 10
0', 'Day 64 of 100', 'Day 65 of 100', 'Day 66 of 100', 'Day 67 of 100', 'Day 68 of 10
0', 'Day 69 of 100', 'Day 7 of 100', 'Day 70 of 100', 'Day 71 of 100', 'Day 72 of 10
0', 'Day 73 of 100', 'Day 74 of 100', 'Day 75 of 100', 'Day 76 of 100', 'Day 77 of 10
0', 'Day 78 of 100', 'Day 79 of 100', 'Day 8 of 100', 'Day 80 of 100', 'Day 81 of 10
0', 'Day 82 of 100', 'Day 83 of 100', 'Day 84 of 100', 'Day 85 of 100', 'Day 86 of 10
0', 'Day 87 of 100', 'Day 88 of 100', 'Day 89 of 100', 'Day 9 of 100', 'Day 90 of 10
0', 'Day 91 of 100', 'Day 92 of 100', 'Day 93 of 100', 'Day 94 of 100', 'Day 95 of 10
```

In [47]:
```python
print(os.getcwd())
#os.chdir('/Users')
#print(os.getcwd())
# google how to remove a file using os
```

```
C:\Users\suman\PYTHON_FOLDER\Tutorials_CodeWithHarry\100_days_python_course
```

#47 Exercise 4 Solution

In [48]:
```python
# Do it after doing the exercise
```

#48 Local vs Global variables

Local - defined inside a function. Global - defined separately.

In [49]:
```
x = 5
print('global variable, x =', x)

def day48():
    x = 50
    y = 45
    print(f'x = {x} and y = {y} are local variables')
day48()

print(x)
try:
    print(y)
except Exception as e:
    print(e)

def day48f():
    global x
    x = 15
    print('global x is changed')
day48f()
print(f'x = {x}')
```

```
global variable, x = 5
x = 50 and y = 45 are local variables
5
name 'y' is not defined
global x is changed
x = 15
```

#49 file IO

In [50]:
```
file1 = open('file1.txt', 'r') # 'r' for read mode (default)
print(file1)
text1 = file1.read()
file1.close()
print(text1)
print(type(file1), type(text1))
```

```
<_io.TextIOWrapper name='file1.txt' mode='r' encoding='cp1252'>
Hello! This file is made for the tutorial 49 of the 100 days pyhton course by CodeWithHarr
y.
We have created this in the folder for the mentioned course.
We are going to open this file in python.

<class '_io.TextIOWrapper'> <class 'str'>
```

In [51]:
```
# file1 = open('file1.txt', 'a') # append mode
# file1.writelines('This line is added by code.')
```

In [52]:
```
file2 = open('file2.txt', 'w')
file2.writelines('This line is written by python codes.')
file2.write(' This is the second line written in python.')
file2.close() # needed
```

In [53]:
```
file1 = open('file1.txt', 'rb') # read binary; default - read text (rt)
print(file1.read())
```

```
b'Hello! This file is made for the tutorial 49 of the 100 days pyhton course by CodeWithHa
rry.\r\nWe have created this in the folder for the mentioned course.\r\nWe are going to op
en this file in python.\r\n'
```

In [54]:
```
with open('file2.txt', 'a') as fl2: # the file will be automatically closed here
    fl2.write(' code runned.')
```

#50 read(), readlines() and other Methods

In [55]:
```
file1 = open('file1.txt', 'r')
file3 = open('file3.txt', 'r')

while True:
    line = file1.readline()
    if not line:
        print(line, type(line))
        break
    print(line)

i = 0
while True:
    line3 = file3.readline()
    if not line3:
        break
    m1 = float(line3.split(',')[0])
    m2 = float(line3.split(',')[1])
    m3 = float(line3.split(',')[2])
    m4 = float(line3.split(',')[3])
    m5 = float(line3.split(',')[4])
    m = [m1, m2, m3, m4, m5]
    i += 1
    for j in range(5):
        print(f'Student {i} has obtained {m[j]} marks in subject {j+1}.')
    print(line3)
```

```
Hello! This file is made for the tutorial 49 of the 100 days pyhton course by CodeWithHarr
y.

We have created this in the folder for the mentioned course.

We are going to open this file in python.

 <class 'str'>
Student 1 has obtained 34.0 marks in subject 1.
Student 1 has obtained 63.0 marks in subject 2.
Student 1 has obtained 25.0 marks in subject 3.
Student 1 has obtained 36.0 marks in subject 4.
Student 1 has obtained 24.0 marks in subject 5.
34,63,25,36,24

Student 2 has obtained 48.0 marks in subject 1.
Student 2 has obtained 39.0 marks in subject 2.
Student 2 has obtained 35.0 marks in subject 3.
Student 2 has obtained 95.0 marks in subject 4.
Student 2 has obtained 36.0 marks in subject 5.
48,39,35,95,36

Student 3 has obtained 24.0 marks in subject 1.
Student 3 has obtained 85.0 marks in subject 2.
Student 3 has obtained 62.0 marks in subject 3.
Student 3 has obtained 85.0 marks in subject 4.
Student 3 has obtained 43.0 marks in subject 5.
24,85,62,85,43

Student 4 has obtained 52.0 marks in subject 1.
Student 4 has obtained 96.0 marks in subject 2.
Student 4 has obtained 84.0 marks in subject 3.
Student 4 has obtained 50.0 marks in subject 4.
Student 4 has obtained 37.0 marks in subject 5.
52,96,84,50,37
```

In [56]:
```
file2 = open('file2.txt', 'w')
lines = ['line 1\n', 'line2\n'] # \n is needed to have a new line
file2.writelines(lines)
file2.close()
```

In [57]:
```
# roughs
i = 0
while True:
    for j in range(2):
        print(f'Student {i+1} has passed subject {j+1}.')
    i += 1
    if i == 3:
        break
```

```
Student 1 has passed subject 1.
Student 1 has passed subject 2.
Student 2 has passed subject 1.
Student 2 has passed subject 2.
Student 3 has passed subject 1.
Student 3 has passed subject 2.
```

#51 seek(), tell() and other functions

In [58]:
```
with open('file1.txt', 'r') as f:
    print(f.read())
    f.seek(12)
    print(f.tell())
    data1 = f.read(7)
    print(data1)
    print(f.tell())
    print(f.read(5))

with open('file2.txt', 'w') as f:
    f.write('This is a new line.')
    f.truncate(18)
with open('file2.txt', 'r') as f:
    print(f.read())
```

```
Hello! This file is made for the tutorial 49 of the 100 days pyhton course by CodeWithHarr
y.
We have created this in the folder for the mentioned course.
We are going to open this file in python.

12
file is
19
 made
This is a new line
```

#52 lambda functions

By this we can write a function (can be anonymous also) in one line.

In [59]:
```
def squaref(x):
    return x**2
squarelf = lambda x: x**2
print(squaref(15), squarelf(15))

magnitudef = lambda x, y, z: sqrt(x**2+y**2+z**2)
print(magnitudef(12,5,0))

def appl(fx, gx, x):
    return fx(x)**2 +gx(x)**2 - (fx(x)*gx(x))**2
print(appl(lambda x: x**2-2*x, lambda x: x**2+2*x, 2))
```

```
225 225
13.0
64
```

#53 map, filter and reduce

If a function takes another function as an argument, the function is called higher order function. **`map`** , **`filter`** and **`reduce`** are higher order functions.

map: It allows lists as arguments of functions.

filter: It returns a boolean value and is applied on iterable objects.

In [60]:
```
l = range(13)
try:
    print(squaref(l))
except Exception as e:
    print(e)

newl = []
for item in l:
    newl.append(squaref(item))
print(newl)

newlm = map(squaref, l)
print(list(newlm))
```

```
unsupported operand type(s) for ** or pow(): 'range' and 'int'
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144]
```

In [61]:
```
filterfn = lambda a: a>5
newfl = filter(filterfn, l)
print(list(newfl))
```

```
[6, 7, 8, 9, 10, 11, 12]
```

In [62]:
```
from functools import reduce

multf = reduce(lambda x,y: x*y, [3,5,2,6])
print(multf)
```

```
180
```

#54 'is' vs '=='

is: compares exact location of object in memory.

==: compares values.

In [63]:
```python
a, b = 4, '4'
print(a is b, a==b)
a, b = [2,6,2], [2,6,2]
print(a is b, a==b)
a, b = 'skp', 'skp'
print(a is b, a==b) # python has taken same memory location
a, b = (0,1), (0,1)
print(a is b, a==b)
a, b = None, None
print(a is b, a==b)
```

```
False False
False True
True True
True True
True True
```

#55 Exercise 5: Snake Water Gun

1. Snake beats water, water beats gun and gun beats snake.
2. Write a python program to create this game using if-else statements. Use proper functions to check for win.

```python
import random
swg = [1,2,3]

points = 0
print('1 for Snake, 2 for Water, 3 for Gun')

while True:
    comp = random.choice(swg)
    player = int(input('enter 1 for Snake, 2 for Water, 3 for Gun: '))
    print(f'By user: {player} \t By computer: {comp}')
    if player==comp:
        print('Draw!')
        print(f'Points = {points} +0 = {points}')
    elif player-comp==(-1):
        print('You Won!')
        print(f'Points = {points} +1 = {points+1}')
        points += 1
    elif player-comp==1:
        print('You Lose!')
        print(f'Points = {points} -1 = {points-1}')
        points += (-1)
    elif player-comp==2:
        print('You Won!')
        print(f'Points = {points} +1 = {points+1}')
        points += 1
    elif player-comp==(-2):
        print('You Lose!')
        print(f'Points = {points} -1 = {points-1}')
        points += (-1)
    else:
        print('Wrong input!')
    nxt = input('enter y for another round and enter n for stopping the game: ')
    if nxt == 'y':
        continue
    if nxt == 'n':
        break
```

#56

In [ ]:
```python

```