Open in Colab

chapter 01 - Working with numbers

In [ ]:

```
print(1+ 2)  # addition and subtractions by using the operators
print(1+3.5)  # (+) and (-) operators
print(-1 + 2.5)
print(100 - 45)
- 1.1 + 5
```

```
3
4.5
1.5
55
```

Out[ ]:

```
3.9
```

In [ ]:

```
print(3 * 2 ) # multiplication (* ) operator
print(3.5 * 1.5 )
```

```
6
5.25
```

In [ ]:

```
print(3/2) # division (/ )  using  the opeartor
print(4/2)
```

```
1.5
2.0
```

In [ ]:

```
# results in the  integers form of floor division eliminating  decimal
# using the (// )  operator
print(3//2)
```

```
1
```

In [ ]:

```
print(-3//2)
```

```
-2
```

In [ ]:

```
print(9 % 2 ) # if u want the remainder just use the modulo (% ) operator
```

```
1
```

In [ ]:

```
print(2**2 )   # calculating the power of numbers by using the exponentional
print(2** 10 )
print(1** 10 )
```

```
4
1024
1
```

In [ ]:

```
print( 8 ** (1/3)) # calulating the power less then  < 1
                   #square root of number  n can be expressed as (n(1/2))
                   #cube root of the number n as expressed as (n(1/3))
```

```
2.0
```

In [ ]:

```
# PEMDAS rule : parentheses () , exponents , multiplication , divison ,additi
print(5+5 * 5 )
(5+5) * 5 # using the parantheses without print func
```

```
30
```

Out[ ]:

```
50
```

In [ ]:

```
a = 3  # a is the variable  3 is the value assing to a
a+ 1  # 1 is assigning to the varaible a (a = 3) +1
```

Out[ ]:

```
4
```

In [ ]:

```
a = 5 #example
a+ 4
```

Out[ ]:

```
9
```

In [ ]:

```
print(type(3)) # display the type of the input  ( #integer )
print(type (7.8))  # float point decimal
print(type("rahul"))  # striing
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

In [ ]:

```
type(8.9) , type (9) , type ('rahul')
```

Out[ ]:

```
(float, int, str)
```

In [ ]:

```
# 3.0 , 4.0 python accept it as a folat points   but
# as an integers convert them floating points to integers func used
int(3.8) , int(1.0) , int(3.0) # using the func "int()"
```

Out[ ]:

```
(3, 1, 3)
```

In [ ]:

```
# float() similarly the revers function that used for converting
# the  integers to the floating point decimals
float(3) , float(1) , float(8)
```

Out[ ]:

```
(3.0, 1.0, 8.0)
```

In [ ]:

```
# working with FRACTIONS
from fractions import Fraction # importing fractions module
f = Fraction (3, 4) # (numarator , denaminator )
```

```
f
```

Out[ ]:
```
Fraction(3, 4)
```

In [ ]:
```
Fraction (3, 4 ) +1 + 1.5 # combining the fraction and integer and floating p
                          # When you have a floating point number in an expre
                          #the expression is returned as a floating point num
```

Out[ ]:
```
3.25
```

In [ ]:
```
Fraction (3, 4) +2+ Fraction(2,3) #d, when you have only a fraction and an in
                                  #expression, the result is a fraction, even
```

Out[ ]:
```
Fraction(41, 12)
```

In [ ]:
```
# complex number
a = 2 + 3j
type(a)
```

Out[ ]:
```
complex
```

In [ ]:
```
a = complex(2,3) # define using the complex func
a
```

Out[ ]:
```
(2+3j)
```

In [ ]:
```
b = 3 + 3j
a + b  # a (2 + 3j ) assigns the values
```

Out[ ]:
```
(5+6j)
```

In [ ]:
```
a - b
```

Out[ ]:
```
(-1+0j)
```

In [ ]:
```
a* b
```

Out[ ]:
```
(-3+15j)
```

In [ ]:
```
a/b
```

Out[ ]:
```
(0.8333333333333334+0.16666666666666666j)
```

In [ ]:
```
# the modules of the (%) and (// ) are not valid for the complex numbers
# retrived can by using the attributes "real " amd "imag"
z = 2 + 3j
z.real
```

```
z.real
```

Out[ ]:
```
2.0
```

In [ ]:
```
z.imag
```

Out[ ]:
```
3.0
```

In [ ]:
```
z.conjugate() # conjugate of a comples number has the same real part and imag
```

Out[ ]:
```
(2-3j)
```

In [ ]:
```
(z.real ** 2 + z.imag ** 2) ** 0.5 # calculating the magnitude of a complex n
```

Out[ ]:
```
3.605551275463989
```

In [ ]:
```
abs(z) # the abs() func simple way to calculate the magnitude
# cmath complex math
```

Out[ ]:
```
3.605551275463989
```

In [ ]:
```
# GETTING THE USER INPUT
a = input() # input() accept the user input via input func
            # the input stored in the variable 'a'
```

```
1
```

In [ ]:
```
a # result in the string format '1'   single quotes or  double quotes
```

Out[ ]:
```
'1'
```

In [ ]:
```
s1 = 'rahul '
s2 = " rahul .c "
print(s1)
print(s2)
```

```
rahul
 rahul .c
```

In [ ]:
```
a = '1' # converting the string '1' to int
int(a) +1
```

Out[ ]:
```
2
```

In [ ]:
```
float(a )+ 1 # string to float conversion using the float and int funcs
```

Out[ ]:
```
2.0
```

SESSION 2

In [ ]:

```
a = Fraction(input('enter a fraction'))
```

```
enter a fraction3/4
```

In [ ]:

```
a
```

Out[ ]:

```
Fraction(3, 4)
```

In [ ]:

```
print(a) , type(a)
```

```
3/4
```

Out[ ]:

```
(None, fractions.Fraction)
```

In [ ]:

```
#calculating the factors of an integers
def is_factor(a, b ):
  if b % a == 0:
    return True
  else:
    return False
```

In [ ]:

```
is_factor(4, 1024)
```

Out[ ]:

```
True
```

In [ ]:

```
for i in range(1,4):
  print(i)
```

```
1
2
3
```

In [ ]:

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

In [ ]:

```
for i in range(1, 10, 2):
  print(i)
```

```
1
3
5
7
9
```

In [ ]:

```
def factors(b): # finding the factor of an integer
def factors(b): # finding the factor of an integer

  for i in range(1, b+1):
    if b% i == 0:
      print(i)

if __name__ == '__main__':
  b = input('your integer :')
  b = float(b)

  if b> 0 and b.is_integer():
    factors(int(b))
  else:
    print('please enter a positive integer')
```

```
your integer :4
1
2
4
```

In [ ]:

```
def factors(b): # finding the factor of an integer

  for i in range(1, b+1):
    if b% i == 0:
      print(i)

if __name__ == '__main__':
  b = input('your integer :')
  b = float(b)

  if b> 0 and b.is_integer():
    factors(int(b))
  else:
    print('please enter a positive integer')
```

```
your integer :4.5
please enter a positive integer
```

In [ ]:

```
item1 = 'apples'
item2 = 'bananas'
item3 = 'grapes'
print('at the grocery store , i bought some{0} and {1} and {2}'. format(item1
```

```
at the grocery store , i bought someapples and bananas and grapes
```

SESSION 3

In [ ]:

```
print('number 1: {0} number 2 :{1}'.format(1,3.578))
```

```
number 1: 1 number 2 :3.578
```

In [ ]:

```
def multiple_table(a):   #multiplcation table printer  format() method.

  for i in range(1, 11):
    print('{0} x {1} = 2'.format(a, i, a*i))


if __name__=='__main__':
  a = input('enter a number : ')
  multiple_table(float(a))
```

```
enter a number : 5
5.0 x 1 = 2
5.0 x 2 = 2
5.0 x 3 = 2
5.0 x 4 = 2
5.0 x 5 = 2
5.0 x 6 = 2
5.0 x 7 = 2
5.0 x 8 = 2
5.0 x 9 = 2
5.0 x 10 = 2
```

In [ ]:

```
'{0}'.format(1.25456) , '{0:.2f}'.format(1.25456)
```

Out[ ]:

```
('1.25456', '1.25')
```

In [ ]:

```
'{0:.2f}'.format(1.25556)
```

Out[ ]:

```
'1.26'
```

In [ ]:

```
'{0:.2f}'.format(1)
```

Out[ ]:

```
'1.00'
```

In [ ]:

```
(25.5 * 2.54) / 100 # converting units of measurements
```

Out[ ]:

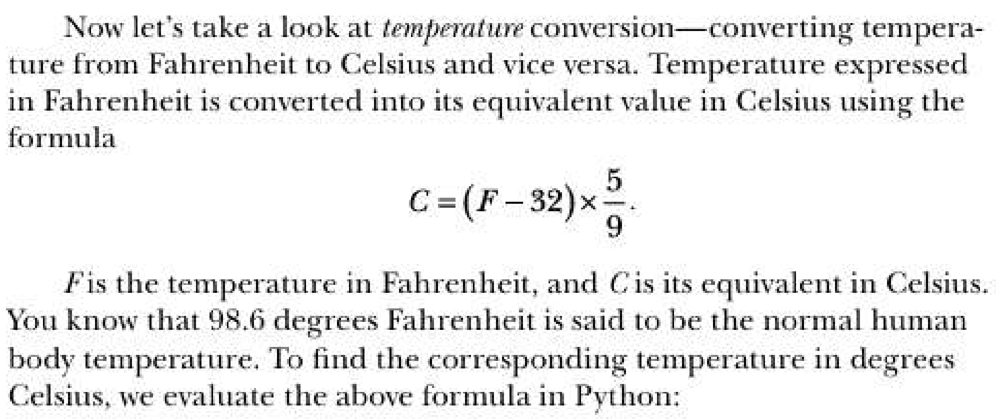
```
0.6476999999999999
```

In [ ]:

```
650 *1.609
```

Out[ ]:

```
1045.85
```



Now let's take a look at *temperature* conversion—converting temperature from Fahrenheit to Celsius and vice versa. Temperature expressed in Fahrenheit is converted into its equivalent value in Celsius using the formula

$$C = (F - 32) \times \frac{5}{9}.$$

F is the temperature in Fahrenheit, and *C* is its equivalent in Celsius. You know that 98.6 degrees Fahrenheit is said to be the normal human body temperature. To find the corresponding temperature in degrees Celsius, we evaluate the above formula in Python:

In [ ]:

```
F = 98.6  #converting temperature to its equilent in celsius
(F - 32 )*(5/9)
```

Out[ ]: 37.0

In [ ]:

```
c = 37 # converting the celsius to  fahrenheit
c* (9/5) + 32
```

Out[ ]: 98.60000000000001

In [ ]:

```
def print_menu():
  print('1. kilometers to miles ')          # unit conversion ; miles to kilom
  print('2. miles to kilometers ')

def km_miles():
  km = float(input('enter the distance in kilometers : '))
  miles = km / 1.609
  print('distance in miles: {0}'.format(miles))

def miles_km():
  miles = float(input('enter the distance in miles :'))
  km = miles * 1.609
  print('distance in kilometers: {0}'.format(km))

if __name__ == '__main__' :
  print_menu()
  choice = input('which conversion would you like to do?:')
  if choice == '1':
    km_miles()

  if choice  == '2':
    miles_km()
```

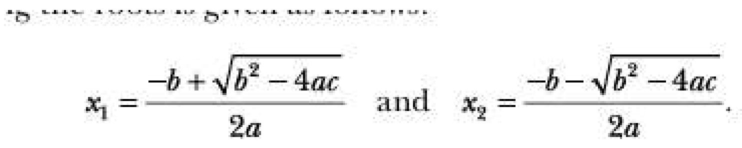
```
1. kilometers to miles
2. miles to kilometers
which conversion would you like to do?:2
enter the distance in miles :100
distance in kilometers: 160.9
```

Finding the Roots of a Quadratic Equation

In [ ]:

```
x = 10 - 500 + 79
x
```

Out[ ]: -411



g the roots is given as follows:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

In [ ]:

```
a =1
```

```
b = 2
c = 1
d = (b**2 - 4*a*c) ** 0.5
x_1 = (-b + d)/(2*a)
x_1
```

Out[ ]:

```
-1.0
```

In [ ]:

```
x_2 = (-b + d)/(2* a)
x_2
```

Out[ ]:

```
-1.0
```

Quadratice equation root calculator

In [ ]:

```
def roots(a, b , c):
  d = (b*b - 4*a*c) ** 0.5
  x_1 = (-b + d)/(2*a)
  x_2 = (-b - d)/(2*a)

  print('x1: {0}'.format(x_1))
  print('x2: {0}'.format(x_2))

if __name__ == '__main__':
  a = input('enter a:')
  b = input('enter b:')
  c = input('enter c:')
  roots(float(a), float(b), float(c)) # a =1 , b = 2 , c = 1
```

```
enter a:1
enter b:2
enter c:1
x1: -1.0
x2: -1.0
```

In [ ]:

```
def roots(a, b , c):
  d = (b*b - 4*a*c) ** 0.5
  x_1 = (-b + d)/(2*a)
  x_2 = (-b - d)/(2*a)

  print('x1: {0}'.format(x_1))
  print('x2: {0}'.format(x_2))

if __name__ == '__main__':
  a = input('enter a:')
  b = input('enter b:')
  c = input('enter c:')
  roots(float(a), float(b), float(c)) # a =1 , b = 1 , c = 1

print('the roots printed above are complex numbers indiicating the j ')
```

```
enter a:1
enter b:1
enter c:1
x1: (-0.49999999999999994+0.8660254037844386j)
x2: (-0.5-0.8660254037844386j)
```

```
the roots printed above are complex numbers indiicating the j
```

programming challenges

In [ ]:

```python
from fractions import Fraction

def add(a,b):
  print('result of addition: {0}'.format(a+b))    # fractions operations


if __name__ == '__main__':
  a = Fraction(input('enter first fraction:'))
  b = Fraction(input('enter second fraction:'))
  op = input('operation to perform - add , subtract, divide , multiply:')
  if op == 'add':
    add(a,b)
```

```
enter first fraction:3/4
enter second fraction:1/4
operation to perform - add , subtract, divide , multiply:add
result of addition: 1
```

In [ ]:

```python
from fractions import Fraction

def add(a, b):
    print('Result of addition: {0}'.format(a + b))

def subtract(a, b):
    print('Result of subtraction: {0}'.format(a - b))

def multiply(a, b):
    print('Result of multiplication: {0}'.format(a * b))

def divide(a, b):
    if b != 0:
        print('Result of division: {0}'.format(a / b))
    else:
        print('Error: Cannot divide by zero.')

if __name__ == '__main__':
    a = Fraction(input('Enter first fraction (e.g., 1/2): '))
    b = Fraction(input('Enter second fraction (e.g., 1/3): '))
    op = input('Operation to perform - add, subtract, divide, multiply: ')

    if op == 'add':
        add(a, b)
    elif op == 'subtract':
        subtract(a, b)
    elif op == 'multiply':
        multiply(a, b)
    elif op == 'divide':
        divide(a, b)
    else:
        print('Invalid operation. Please choose from add, subtract, divide, m
```

```
Enter first fraction (e.g., 1/2): 3/4
Enter first fraction (e.g., 1/2): 3/4
Enter second fraction (e.g., 1/3): 1/4
Operation to perform - add, subtract, divide, multiply: subtract
Result of subtraction: 1/2
```

In [ ]:

```
def fun():
  print('iam in an endless loop ') # run untill exit layout infinate loop


if __name__ == '__main__':
  while True:
    fun()
    answer = input('do u want to exist? (y) for yes : ')
    if answer == 'y':
      break
```

```
iam in an endless loop
do u want to exist? (y) for yes : h
iam in an endless loop
do u want to exist? (y) for yes : j
iam in an endless loop
do u want to exist? (y) for yes : k
iam in an endless loop
do u want to exist? (y) for yes : i
iam in an endless loop
do u want to exist? (y) for yes : y
```

In [ ]:

```
def multi_table(a): # multiplication table printer with the exit power to the

  for i in range(1,11):
    print('{0} x {1} = {2}'.format(a, i, a*i))

if __name__ == '__main__':
  while True:
    a = input('enter a number :')
    multi_table(float(a))

    answer = input('do u want to exist ?(y) for yes :')
    if answer == "y" :
      break
```

```
enter a number :6
6.0 x 1 = 6.0
6.0 x 2 = 12.0
6.0 x 3 = 18.0
6.0 x 4 = 24.0
```