0.1 Exercise 1

[1]: import math

[2]: def isNumber(x):
    
    """
    Returns True if x is int or float
    """
    if type(x) is not int and type(x) is not float:
        return False
    return True

[3]: # Hint: check your code with some examples using 
# https://www.mathsisfun.com/quadratic-equation-solver.html

def quadratic(a, b, c):
    """
    Returns the two solutions to the quadratic equation a*x^2+b*x+c=0
    Expects the three inputs to be int or float
    If the input parameters are invalid, will give an error message and return -999,-999
    If the solution involves a sqrt of a negative number, we will return complex numbers
    For two real solutions, it will return the two solutions ordered as largest,smallest
    """

    # check the three inputs
    if not isNumber(a):
        print('a = ',a,' is not a number. Returning -999,-999')
        return -999,-999
    if not isNumber(b):
        print('b = ',b,' is not a number. Returning -999,-999')
        return -999,-999
    if not isNumber(c):
        print('c = ',c,' is not a number. Returning -999,-999')
        return -999,-999

    # if a is zero, this is not really a quadratic, but let's solve it anyway
if a == 0:
    if b == 0:
        print('Cannot have both a=0 and b=0. Returning -999,-999')
        return -999,-999
    else:
        return -c/b,-c/b

# solution is (-b +/- sqrt(b^2-4ac))/(2a)
temp = b*b - 4*a*c
if temp >= 0:
    x1 = (-b + math.sqrt(temp))/(2*a)
    x2 = (-b - math.sqrt(temp))/(2*a)
    return max(x1,x2),min(x1,x2)
else:
    x1 = complex(-b, math.sqrt(-temp))/(2*a)
    x2 = complex(-b, -math.sqrt(-temp))/(2*a)
    return x1,x2

# Height reached by a ball as a function of time
# h = -1/2 g t^2 + v_0 t
# use SI units

g  = 9.8
v0 = 20

# The heights we want to solve for
heights = (10, 15, 25)

# Solve
# If the solution is real, we want the smallest one.
# This is because the smallest solution will give us the time
# that the ball reaches the desired height on the way up, the
# other solution will give the time on the way down
# Note: since we give g to 0.1 out of 9.8, ie, to about 1%,
# no point in reporting a solution with too many decimal digits
for h in heights:
    t1,t2 = quadratic(-g/2, v0, -h)
    if isNumber(t1):
        print('It will take time t = %.2f sec to reach height h = %.1f meters' % (t2,h))
    else:
        print('The ball will never reach height h = %.1f meters' % h)

It will take time t = 0.58 sec to reach height h = 10.0 meters
It will take time t = 0.99 sec to reach height h = 15.0 meters
The ball will never reach height h = 25.0 meters
0.2 Exercise 2

[5]: # Is this a \geq 0 integer?
def isPosInteger(blah):
    try:
        j = int(blah)
    except:
        return False
    if j<0:
        return False
    return True

# Stirling approximation: \log(n!) = (n+0.5) \times \log(n) - n + 0.5\log(2 \pi)
def stirling(num):
    """
    Returns stirling approx for the \log(num)
    If num not a \geq 0 integer, give error and return zero
    """
    if not isPosInteger(num):
        print("Illegal call to stirling function. Input = ",num)
        return 0
    if num==0:
        return 0
    return (num+0.5)*math.log(num) - num + 0.5*math.log(2 * math.pi)

# Factorial function
def factorial(n):
    """
    Return n factorial.
    Returns 0 if n not integer \geq 0
    """
    if not isPosInteger(n):
        print("Illegal call to factorial function. Input = ",n)
        return 0
    f = 1
    for k in range(1,n+1):
        f = f*k
    return f

[ ]: # We will repeat a few times to show different results
    for i in range(5):
        print('--------------')
        noGoodInput = True
        while noGoodInput:
            number = input("Enter an integer")
```python
# check input
if not isPosInteger(number):
    print(number, " is not an integer >= 0 .... Try again")
    continue

# OK, we have a valid input
noGoodInput = False
n = int(number)
fact = factorial(n)

if fact < 1e6:
    print(str(n)+"! = ", fact)
else:
    print(str(n)+"! = %.6e " %fact)
print("Stirling approximation " + str(n) + "! = %.6e " %math.
    .exp(stirling(n)))

--------------
Enter an integer 3rt5
3rt5  is not an integer >= 0 ... Try again
Enter an integer 9,2
9,2  is not an integer >= 0 ... Try again
Enter an integer -2.3
-2.3  is not an integer >= 0 ... Try again
Enter an integer -2
-2  is not an integer >= 0 ... Try again
Enter an integer 67
67! = 3.647111e+94
Stirling approximation 67! = 3.642578e+94
--------------
```

[ ]: