8 / 10 pts

Group_08_exersice_06

February 8, 2021

1 Help Function for tests

[ ]:
```
def assert_equals(a,b):
    return a == b
```

2 Exercice 1

2.5 / 2.5 pts

In John's car the GPS records every s seconds the distance travelled from an origin (distances are measured in an arbitrary but consistent unit). For example, below is part of a record with s = 15:

`x = [0.0, 0.19, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25]`

The sections are:

`0.0-0.19, 0.19-0.5, 0.5-0.75, 0.75-1.0, 1.0-1.25, 1.25-1.50, 1.5-1.75, 1.75-2.0, 2.0-2.25`

We can calculate John's average hourly speed on every section and we get:

`[45.6, 74.4, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0]`

Given s and x the task is to return as an integer the *floor* of the maximum average speed per hour obtained on the sections of x. If x length is less than or equal to 1 return 0 since the car didn't move.

Example: with the above data your function gps(s, x)should return 74

Note With floats it can happen that results depends on the operations order. To calculate hourly speed you can use:

`(3600 * delta_distance) / s.`

[ ]:
```
# your code here
def gps(s, x):
    new =[]
    if len(x) <=1:
        return 0
    for i in range(0,(len(x)-1)):
        if i < len(x):
            delta= x[i+1]- x[i]
            result=int(3600*delta)/s
```

```
            new.append(result)
            i = i+1
    r=int(max(new))

    return r
```

[ ]:
```
x = [0.0, 0.19, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25]
gps(15,x)
```

[ ]:
```
# Basic Tests
x = [0.0, 0.23, 0.46, 0.69, 0.92, 1.15, 1.38, 1.61]
s = 20
u = 41
print(assert_equals(gps(s,x),u))
################################################################
x = [0.0, 0.11, 0.22, 0.33, 0.44, 0.65, 1.08, 1.26, 1.68, 1.89, 2.1, 2.31, 2.
↪52, 3.25]
s = 12
u = 219
print(assert_equals(gps(s, x), u))
################################################################
x = [0.0, 0.18, 0.36, 0.54, 0.72, 1.05, 1.26, 1.47, 1.92, 2.16, 2.4, 2.64, 2.
↪88, 3.12, 3.36, 3.6, 3.84]
s = 20
u = 80
print(assert_equals(gps(s, x), u))
################################################################
x = [0.0, 0.02, 0.36, 0.54, 0.72, 0.9, 1.08, 1.26, 1.44, 1.62, 1.8]
s = 17
u = 72
print(assert_equals(gps(s, x), u))
################################################################
x = [0.0]
s = 19
u = 0
print(assert_equals(gps(s, x), u))
################################################################
x = []
s = 19
u = 0
print(assert_equals(gps(s, x), u))
```

3 Exercice 2

1.5 / 2.5 pts

In mathematics, the factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n. For example:

`5! = 5 * 4 * 3 * 2 * 1 = 120`

By convention the value of `0! is 1`. Write a function to calculate factorial for a given input. If input is below 0 or above 12 raise an exception of type ValueError (Python).

[ ]:
```
# your code here
def factorial(nb):
    if nb == 0:
        return (1)
    if nb>= 1:
        return nb * (factorial(nb-1))
```

[ ]:
```
print(assert_equals(factorial(0),1))
print(assert_equals(factorial(1),1))
print(assert_equals(factorial(2),2))
print(assert_equals(factorial(3),6))
print(assert_equals(factorial(4),24))
print(assert_equals(factorial(6),720))
```

3.1 Exercice 3

1 / 2 pts

Try to write the exercice 6.1 again but without using a `for` or a `while` loop. Just a **recursion**.

[ ]:
```
# your code here
def factorial(nb):
    factorial = 1
    if nb == 0:
        return factorial
    if int(nb) >= 1:
        for i in range (1,int(nb)+1):
            factorial = factorial * i
        return factorial
```

[ ]:
```
print(assert_equals(factorial(0),1))
print(assert_equals(factorial(1),1))
print(assert_equals(factorial(2),2))
print(assert_equals(factorial(3),6))
print(assert_equals(factorial(4),24))
print(assert_equals(factorial(6),720))
```

4 Exercice 4

3 / 3 pts

Write a recursiv funciton called `step_sum`, die a function that reduces an array of integer numbers by adding adjacent elements until only a single element is left. The reduced list should be output step by step. Example: For the list `l = [2, 4, 1, 3, 7]` the following output is to be generated:

‘

43 20 23 11 9 14 6 5 4 10 2 4 1 3 7 ‘

Line 5 is the array. In line 4, $2 + 4 = 6$, $4 + 1 = 5$, $1 + 3 = 4$ and $3 + 7 = 10$ are computed und displayd, etc.

[1]:
```
def step_sum(l):
    total =0
    newlist=[]
    for i in range (0,(len(l)-1)):
        total= l[i] + l[i+1]
        newlist.append(total)
        i = i+1
    print(newlist)
    if len(newlist)>1:
        step_sum(newlist)



list1=[2, 4, 1, 3, 7]
step_sum(list1)
```

```
[6, 5, 4, 10]
[11, 9, 14]
[20, 23]
[43]
```

[ ]:
```

```