ENGR 1330-2022-1 Exam1-Laboratory Portion

Medrano, Giovanni

R11521018

ENGR 1330 Exam 1 - Laboratory/Programming Skills

Problem 1 (10 pts) : *Profile your computer*

Execute the code cell below exactly as written. If you get an error just continue to the remaining problems.

In [1]:
```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Problem 2 (10 pts): *input(),typecast, string reversal, comparison based selection, print()*

Build a script where the user will supply a number then determine if it is a palindrome number. A palindrome number is a number that is same after reversal. For example 545, is a palindrome number.

- Case 1: 545
- Case 2: 123
- Case 3: 666

In [ ]:
```
# define variables
# interactive input
# computation/compare
# report result
```

In [1]:
```
# Case 1
num = 545
num = str(num) # '545'
length = len(num)
```

```
position = length - 1 # end the string
rev = ''          # blank variable for reversed string
for i in range(length):
    rev += num[position] # begin iteration within num at position position which we def
    position -=  1   # iterating backwards and update
if(rev == num):
    print(num, 'is a palindrone!')
else:
    print(num, 'is not a palindrone!')
```

```
545 is a palindrone!
```

In [5]:

```
# Case 2
num = 123
num = str(num)
length = len(num)
position = length - 1 # end the string
rev = ''          # blank variable for reversed string
for i in range(length):
    rev += num[position] # begin iteration within num at position position which we def
    position -=  1   # iterating backwards and update
if(rev == num):
    print(num, 'is a palindrone!')
else:
    print(num, 'is not a palindrone!')
```

```
123 is not a palindrone!
```

In [6]:

```
# Case 3
num = 666
num = str(num)
length = len(num)
position = length - 1 # end the string
rev = ''          # blank variable for reversed string
for i in range(length):
    rev += num[position] # begin iteration within num at position position which we def
    position -=  1   # iterating backwards and update
if(rev == num):
    print(num, 'is a palindrone!')
else:
    print(num, 'is not a palindrone!')
```

```
666 is a palindrone!
```

In [7]:

```
num = input("Please input a number to check for palindrome-ness:")
rev = num[::-1]
if(rev == num):
    print(num, 'is a palindrone!')
else:
    print(num, 'is not a palindrone!')
```

```
166858661 is a palindrone!
```

Problem 3 (15 pts): *len(),compare,accumulator, populate an empty list,for loop, print()*

Two lists are defined as

```
x= [1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8]
f_of_x = [1.543,1.668,1.811,1.971,2.151,2.352,2.577,2.828,3.107]
```

Create a script that determines the length of each list and if they are the same length then print the contents of each list row-wise, and the running sum of `f_of_x` so the output looks like

```
--x--  --f_of_x--  --sum--
1.0     1.543      1.543
1.1     1.668      3.211
...     ...        ...
...     ...        ...
1.7     2.828      16.901
1.8     3.107      20.008
```

Test your script using the two lists above, then with the two lists below:

```
x= [1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8]
f_of_x =[1.543, 3.211, 5.022, 6.993, 9.144, 11.496, 14.073, 16.901,
20.008]
```

In [90]:

```
# define variables
# Case 1
x = [1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8]
f_of_x = [1.543,1.668,1.811,1.971,2.151,2.352,2.577,2.828,3.107]
# validate lengths
lenX = len(x)
lenF = len(f_of_x)
#print(lenX, lenF)
# initialize accumulator and empty list to store a running sum
if(lenX == lenF):
    total = []
# print header line
    print("--x--  --f_of_x--  --sum--")
# repetition (for loop) structure
    sums = 0
    for i in range(lenF):
        sums += f_of_x[i]
        sums = round(sums,3)
        total.append(sums)
#print(total)
# report result
    for i in range(lenX):
        print(str(x[i]) + '\t ' + str(f_of_x[i]) + '\t    ' + str(total[i])) #casted to
                                                                            #concatenat
```

```
--x--  --f_of_x--  --sum--
1.0      1.543      1.543
1.1      1.668      3.211
1.2      1.811      5.022
1.3      1.971      6.993
1.4      2.151      9.144
1.5      2.352      11.496
1.6      2.577      14.073
1.7      2.828      16.901
1.8      3.107      20.008
```

In [93]:

```
# define variables
# Case 2
x= [1.0,1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8]
f_of_x =[1.543, 3.211, 5.022, 6.993, 9.144, 11.496, 14.073, 16.901, 20.008]
# validate lengths
lenX = len(x)
lenF = len(f_of_x)
print(lenX, lenF)
# initialize accumulator and empty list to store a running sum
if(lenX == lenF):
    total = []
# print header line
    print("--x--  --f_of_x--  --sum--")
# repetition (for loop) structure
    sums = 0
    for i in range(lenF):
        sums += f_of_x[i]
        sums = round(sums,3)
        total.append(sums)
#print(total)
# report result
    for i in range(lenX):
        print(str(x[i]) + '\t' + str(f_of_x[i]) + '\t   ' + str(total[i]))
```

```
9 9
--x--  --f_of_x--  --sum--
1.0     1.543      1.543
1.1     3.211      4.754
1.2     5.022      9.776
1.3     6.993      16.769
1.4     9.144      25.913
1.5     11.496     37.409
1.6     14.073     51.482
1.7     16.901     68.383
1.8     20.008     88.391
```

Problem 4 Function (15 points) : *def ..., input(),typecast,arithmetic based selection, print()*

Build a function that takes as input two integer numbers. The function should return their product if the product is greater than 666, otherwise the function should return their sum.

Employ the function in an interactive script and test the following cases:

- Case 1: 65 and 10
- Case 2: 66 and 11
- Case 3: 25 and 5

In [ ]:
```
# define variables
# interactive input
# computation/compare
# report result
```

In [19]:
```
# Case 1
a = 65
b = 10
```

```
product = a * b
total = a + b
if(product > 666):
    print('The product is:', product)
else:
    print('Their sum is:', total)
```

```
Their sum is: 75
```

In [20]:
```
# Case 2
a = 66
b = 11
product = a * b
total = a + b
if(product > 666):
    print('The product is:', product)
else:
    print('Their sum is:', total)
```

```
The product is: 726
```

In [21]:
```
# Case 3
a = 25
b = 5
product = a * b
total = a + b
if(product > 666):
    print('The product is:', product)
else:
    print('Their sum is:', total)
```

```
Their sum is: 30
```

In [25]:
```
a, b = input("Enter two integers with a space between them and we will see if their pro
a = float(a) #floats are used incase a user inputs decimal points.
b = float(b)
product = a * b
total = a + b
if(product > 666):
    print('The product is:', product)
else:
    print('Their sum is:', total)
```

```
The product is: 682.5
```

In [ ]: