Pair Programming exercise Functions

DSE5002, Module 7, HD Sheets, July 2024 updated 11/13/2024

Writing functions in Python

see

https://docs.python.org/3/tutorial/controlflow.html#defining-functions

Functions

Function definitions in Python start with the keyword "def" which indicates that we are starting a function definition

After def, we state the name of the function and then the input variables

Here is a function that computes the squares of inters up to n

The input is n and there is no returned variable

The section inside the triple quotes """ is called the docstring it should explain what the function does and what the output is

In [14]:
```
#define the function

def squares2n(n):
    """
    squares2n(n) prints the squares of all integers such that the square is
    less than n,  starting from 1
    """
    x=1
    while(x**2<n):              # notice the use of a while loop here
        print(x,x**2)
        x=x+1
```

now run it

In [15]:
```
squares2n(1500)
```

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
14 196
15 225
16 256
17 289
18 324
19 361
20 400
21 441
22 484
23 529
24 576
25 625
26 676
27 729
28 784
29 841
30 900
31 961
32 1024
33 1089
34 1156
35 1225
36 1296
37 1369
38 1444
```

In [16]:
```
#viewing the docstring, another way to learn about what a function does

print(squares2n.__doc__)
```

```
    squares2n(n) prints the squares of all integers such that the square is
    less than n,  starting from 1
```

Different organizations may have different protocols for what belongs in a docstring and how it should be structured

Question/Action

Write a function in the cell below that takes in two values, a and b, and prints out the value of the smaller of the two

Include a simple doc string

use an if-else pair to do this, you may need to look up if/else in python

In [29]:
```
import random
```

In [30]:
```
def get_smaller(a, b):
    """
    get_smaller takes in two variables and prints or returns the smaller of the two
    """
    smaller = a if a<b else b
    print(f'a={a}\t b={b}\t smaller={smaller}')
    return smaller

for i in range(10):
    get_smaller(random.randrange(0,1000,1),random.randrange(0,1000,1))
```

```
a=224    b=662   smaller=224
a=72     b=510   smaller=72
a=150    b=676   smaller=150
a=934    b=941   smaller=934
a=417    b=954   smaller=417
a=618    b=18    smaller=18
a=809    b=760   smaller=760
a=34     b=913   smaller=34
a=291    b=211   smaller=211
a=319    b=491   smaller=319
```

Return values

A function can return values, just as it can in R

Here, we will alter squares2n to return a list

In [31]:
```
#define the function

def squares2nlist(n):
    """
    squares2n(n) prints the squares of all integers such that the square is
    less than n,  starting from 1

    returns a list of the squares
    """
    y=[]
    x=1
    while(x**2<n):
        y.append(x**2)
        x=x+1
    return y
```

In [32]:
```
#example call
```

```
a=squares2nlist(560)

a
```

Out[32]:
```
[1,
 4,
 9,
 16,
 25,
 36,
 49,
 64,
 81,
 100,
 121,
 144,
 169,
 196,
 225,
 256,
 289,
 324,
 361,
 400,
 441,
 484,
 529]
```

default values on inputs

As in R, we can define default values

In [33]:
```
def powers2n(n=100, power=2):
    """
    powers2n(n,power) computes the powers of the integers less than n and returns t

    inputs are the value n to stop at and the power used

    defaults are n=100, power=2
    """

    y=[]
    x=1
    while(x**power<n):
        y.append(x**power)
        x=x+1
    return y
```

In [34]:
```
a=powers2n()
a
```

Out[34]:
```
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [35]:
```
#calling the function using input parametes in order
a=powers2n(200,3)
a
```

Out[35]:
```
[1, 8, 27, 64, 125]
```

In [36]:
```
# calling the function using named parameters
a=powers2n(power=2.5)
a
```

Out[36]:
```
[1.0,
 5.656854249492381,
 15.588457268119896,
 32.0,
 55.90169943749474,
 88.18163074019441]
```

For more on functions and options to control input parameters, see

https://docs.python.org/3/tutorial/controlflow.html#defining-functions

Question/Action

Modify your function that takes in a and b so that a defaults to 1 and be defaults to 0

Alter the function so that it returns the smaller of the two input values

In [37]:
```
def get_smaller_modified(a=1, b=0):
    """
    get_smaller takes in two variables and prints or returns the smaller of the two
    """
    smaller = a if a<b else b
    print(f'a={a}\t b={b}\t smaller={smaller}')
    return smaller

#Check to see that default is correct
get_smaller_modified()

for i in range(10):
    get_smaller_modified(random.randrange(0,1000,1),random.randrange(0,1000,1))
```

```
a=1      b=0     smaller=0
a=440    b=26    smaller=26
a=702    b=567   smaller=567
a=915    b=910   smaller=910
a=986    b=512   smaller=512
a=914    b=470   smaller=470
a=366    b=331   smaller=331
a=879    b=333   smaller=333
a=531    b=372   smaller=372
a=498    b=558   smaller=498
a=29     b=661   smaller=29
```

Lambda Functions

these are simple, one line functions

They are useful when you need to pass a function or operation into another functions, say if we want to apply the same function all the values along columns of a matrix

In [38]:
```
#creating a function that creates a decrementor function
# this is a function that returns a function which decrements the input by n
# this is the first time we have seen a function that returns a function


def make_decrementor(n):
    return lambda x:x-n
```

In [39]:
```
# make a call to create the decrementor

my_decrement_by_1=make_decrementor(1)

x=10

# my_decrement_by_1 is now a function that decreases it's input by 1

my_decrement_by_1(x)
```

Out[39]: 9

Question/Action

write a function that creates a function that multiples by n

show that it works

In [40]:
```
def make_multiplier(n):
    return lambda x:x*n
```

In [46]:
```
multiply=make_multiplier(2)

x=2

for i in range(31):
    x = multiply(x)
    print(x)
```

```
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
16777216
33554432
67108864
134217728
268435456
536870912
1073741824
2147483648
4294967296
```

Lambda functions are helpful for passing a function into another function

this example allows use to sort by a specific entry in a list of lists

In [47]:
```
customers=[(1,"Lin Ho", "Zhang"),(2,"Smith","Bob"),(3,"Fernandes","Rita")]

#sort by ID

customers.sort(key=lambda x:x[0])
customers
```

Out[47]:
```
[(1, 'Lin Ho', 'Zhang'), (2, 'Smith', 'Bob'), (3, 'Fernandes', 'Rita')]
```

In [48]:
```
# alter this lambda function to sort by the first name
```

In [5]:
```
customers=[(1,"Lin Ho", "Zhang"),(2,"Smith","Bob"),(3,"Fernandes","Rita")]

#sort by ID
```

```
customers.sort(key=lambda x:x[2])
customers
```

Out[5]: [(2, 'Smith', 'Bob'), (3, 'Fernandes', 'Rita'), (1, 'Lin Ho', 'Zhang')]

In [ ]: