L#15-User-Defined Functions

Ago 2019

Where are the library functions required for this course:

	LIBRARY (*)			your own effort
	(1) built-in functions	(2) math module	(3) numpy module	user-defined
import module:	not required	math	numpy	own-module name
how many, are there?	+/- 64	+/- 44	many	$10^{1000000000000}$
Some examples:	abs(), input(), max(), min(), sum(), round(), print()	math.sqrt(x), math.exp(x), math.log(x), math.log10(x), math.sin(x), math.cos(x),	numpy.sin(), numpy.cos(), numpy.radians(),	you can develop as many as you want with your own chosen names

(1) https://docs.python.org/3/library/functions.html (2) https://www.programiz.com/python-programming/modules/math or https://docs.python.org/3/library/math.html (3) https://numpy.org/devdocs/reference/arrays.ndarray.html

Construct a function

Program with no functions

```
x=int(input('Enter the first values '))
y=int(input('Enter the second values '))
z=int(input('Enter the third values '))

mini=x
if (y<mini):
   mini=y
if(z<mini):
   mini=z

print("The minimum value is", mini)
```

Using the logic of a program to construct a function

Function definition

```
def    minimum (x, y, z):
       “’This function finds the minimum of
        3 numbers “’

        mini=x

        if (y<mini):
           mini=y

        if(z<mini):
           mini=z

        return mini
```

A program that finds the minimum of three numbers stored in x, y, z.

A function that finds the minimum of three numbers stored in x, y, z.

A main program with a function within

Open a new file, write the code and save it. For the current case the filename is: function003.py

```
def minimum(x,y,z):
   """ Computes the smallest value of three numbers
   """
   mini=x
   if (y<mini):
      mini=y

   if(z<mini):
      mini=z
   return mini

x=int(input('Enter the first values '))
y=int(input('Enter the second values '))
z=int(input('Enter the third values '))

print(“\nThe minimum value is", minimum(x,y,z))
```

OUTPUT

```
Enter the first values 1

Enter the second values 2

Enter the third values 3

The minimum value is 1
```

OJO: The function definition must be placed before the function call

Array arguments:

x is an array variable with number of elements equal to len(x)

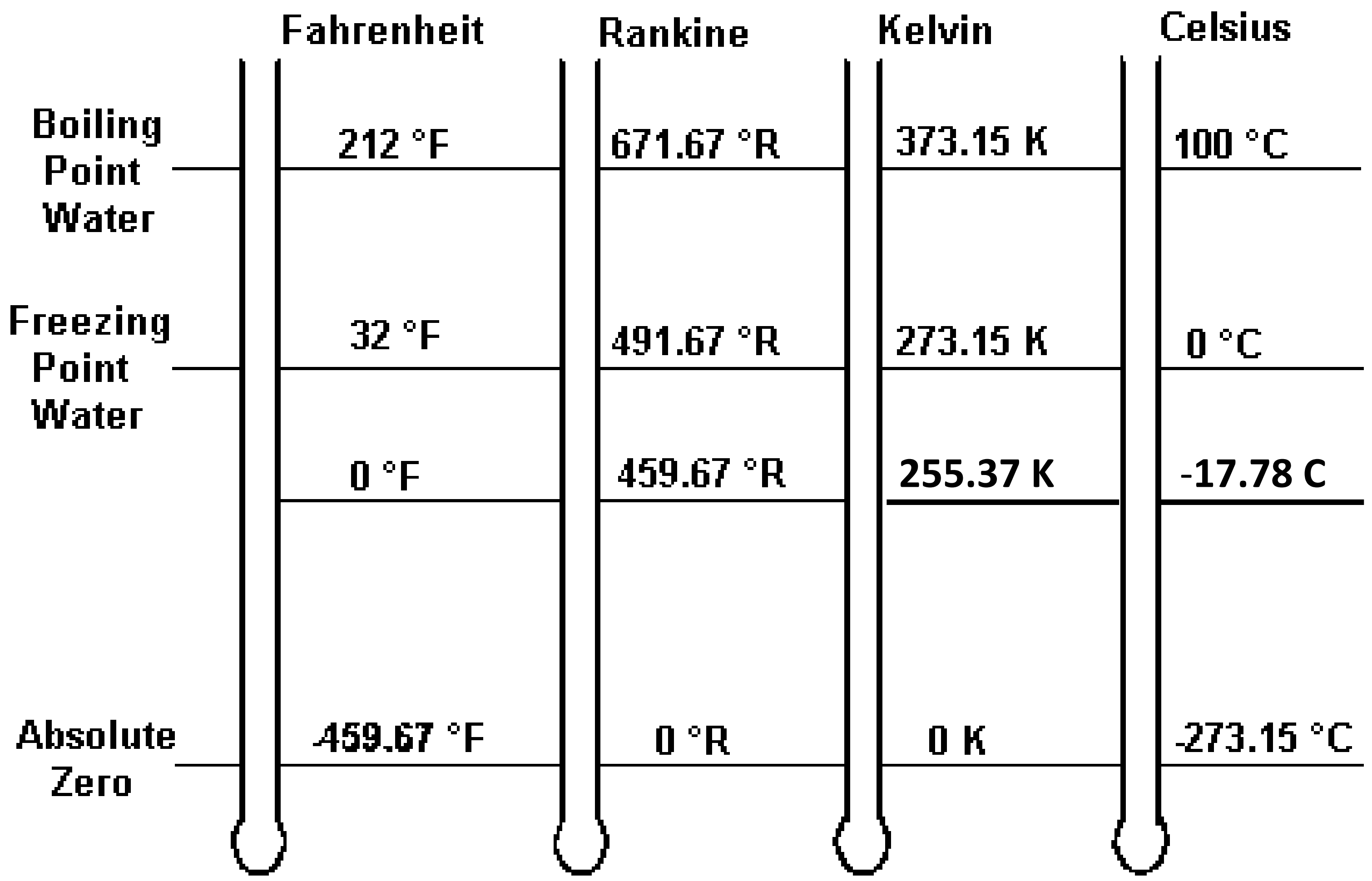
```
1   def minimum2(x):
2          “”””
3          Finds the minimum value
4          in 1D array
5          ”””””
6          mini=x[0]
7          for i in range(1,len(x)):
8              if  x[i]<mini:
9                  mini=x[i]
10         return mini
# i-index in loop is running as
1,2,3,4,….,  length of x
```

```
def minimum2(x):
       “”””
       Finds the minimum value
       in 1D array
       ”””””
       mini=x[0]
       for item  in x:
              if  item<mini:
                mini=item
       return mini
# Line 6 can be substituted by:
#  mini=next(iter(x))
```

```
x =np.array([4,1,0,3,5,8,9,7,2,-1,6])
print( ‘Minimum of set is %d’ , minimum2(x))
```

Quiz: Temperature Conversion Table

Using user-define functions and module. Converts Celsius to Fahrenheit, Rankine, and Kelvin.
Input is C=[-273.25,-17.7777778, 0,100]. i.e., a list with 4 elements. Construct the necessary functions according to table given by instructor. Place these functions into a module called **tempConvert.py**. Then create the main program **tempTable.py** which will import the module containing the user-defined functions called **tempConvert.py**. For bonus points, send PDF report with email subject: UD-Functions and Modules. Team work allowed.



Module

- If you write a function within a program, the function will work only for that program. Functions can be reused if they are placed into a module (i.e., just a python file) and imported into the new program. Once module is imported functions can be called within by dot notation.

```
“””
Module  XXX
My function collection to convert Celsius
Fahrenheit, Kelvin and Rankine
Filename: tempConvert.py “””

def C2F(C)
        bla, bla, bla
def …
        bla, bla, bla
def …
```

```
“””
Main Program
Convert Celsius, Fahrenheit, Rankine, Kelvin
Filename: tempTable.py
“””
import tempConvert as tc

C= something   # data
F=tc.C2F(C)   # function call

OUTPUT
…
…
```

Fibonacci alternatives (1)

```
def fib1(n):
    """ prints Fib up to n """
    a=0; b=1
    while a<n:
        print(a,end=' ')
        c=a+b
        a=b
        b=c
    print()

fib1(10)
fib1(500)
```

```
def fib2(n):
    """ prints Fib up to n """
    a=0; b=1
    while a<n:
        print(a,end=' ')
        c=a+b
        a,b=b,c
    print()

fib2(10)
fib2(500)
```

```
def fib3(n):
    """ prints Fibonacci series up to n """
    a, b=0, 1
    while a<n:
        print(a,end=' ')
        a, b = b, a+b
    print()

fib3(10)
fib3(500)
```

Simultaneously:

$$\begin{tikzcd} b \arrow[d] & a+b \arrow[d] \\ a & b \end{tikzcd}$$

```
OUTPUT
0 1 1 2 3 5 8
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

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

Returned Values: none, one (1)

```
# returns no value
def C2F(C):
    """
    Converts Celcius to Fahrenheit
    """
    print(C,"Celsius =",(9.0/5)*C+32,"Fahrenheit")

C2F(100) #function call
```

OUTPUT

```
100 Celsius = 212.0 Fahrenheit
```

C:\Users\Marco\Python_Programs\multipleReturnFunction.py

```
# return one value
def C2F_1(C):
    """
    Converts Celsius to Fahrenheit
    """
    return (9.0/5.0)*C+32

C=100   # function call is next
print(C,"Celsius =",C2F_1(C),"Fahrenheit")
```

```
100 Celsius = 212.0 Fahrenheit
```

Returned Values: two (2)

```
def C2FK(C):
    """
    Converts Celcius to Fahrenheit and Kelvin
    """
    F=(9.0/5.0)*C+32
    K=C-273.15
    return F,K

C=100
F,K=C2FK(C)    # function call
print(C,"Celcius =",F,"Fahrenheit= %.2f" % K,"Kelvin")
```

```
100 Celcius = 212.0 Fahrenheit= -173.15 Kelvin
```

Returned Values: many

scalar　　array

```
def C2F_1(C):
    """
    Converts C to F
    """
    return (9.0/5.0)*C+32
```

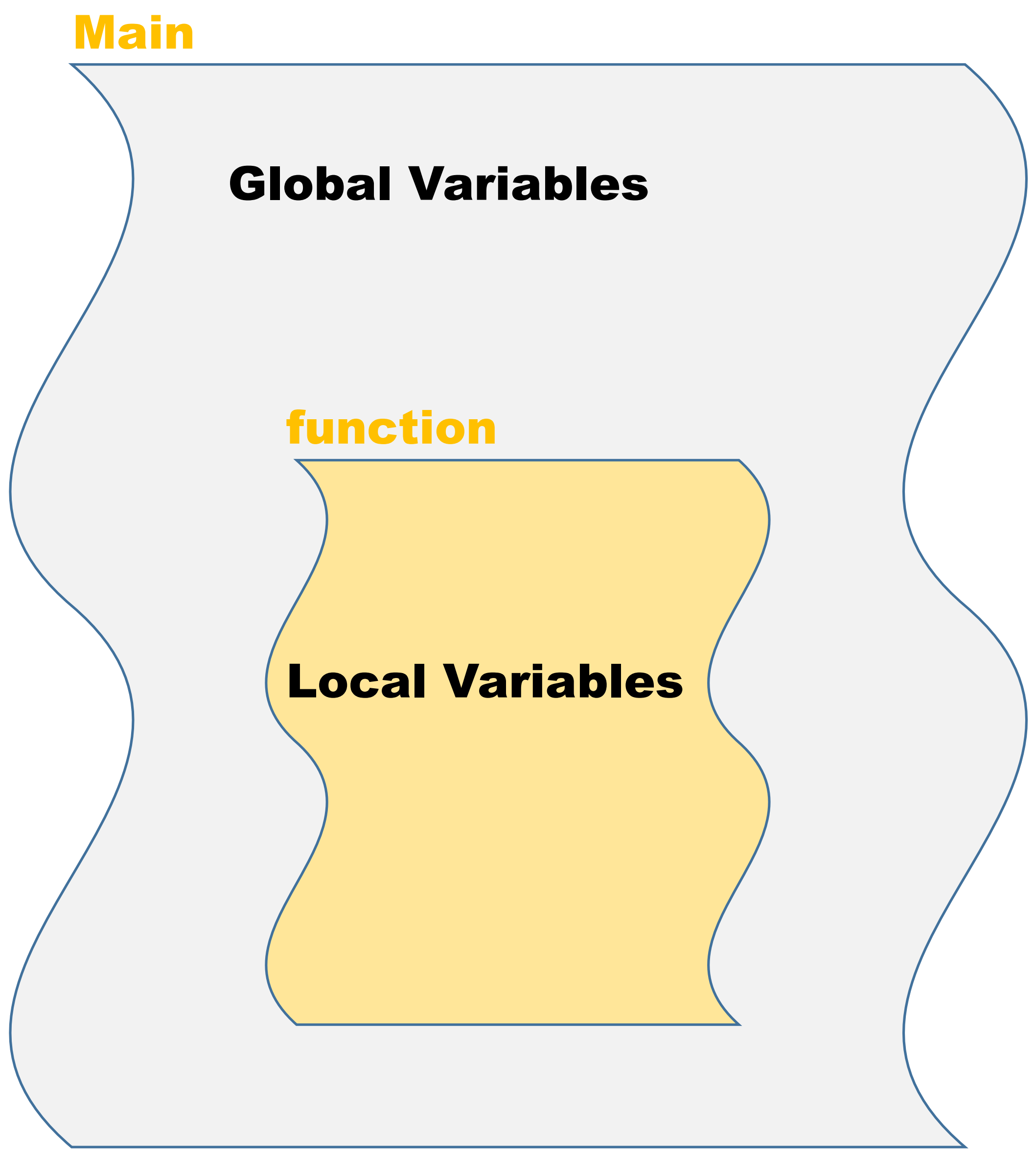
scalar　　array

This function works for scalar or array input
If input is an array C2F_1 produces and array

```
#  return multiple values as sequence
C=np.array([-273.15,-17.7777778,0,100])
F=C2F_1(C)

print("%9s %9s" %("C","F"))
for c,f in zip(C,F):
   print("%9.2f %9.2f" %(c,f))
```

```
        C         F
  -273.15   -459.67
   -17.78     -0.00
     0.00     32.00
   100.00    212.00
```



Variable Scope: Local and Global Variables

Variables can only reach the area in which they are defined, which is called **scope**. Think of it as the area of code where variables can be used.

global variables are usable in the entire program, including outside and inside functions

By default, all variables assigned in a function are **local variables**.

To access a variable defined inside a function outside of it, it's required to explicitly define it as global variable within a function.

Global Variables, Global Scope

```
# Global variables, global scope
s = "I love Algorithms"
x = 5


def f():
    """
    This function uses global
    variable s and x
    """
    print(s)
    print(x)
    return "Have a nice day!!"

print(f())
```

# Variables assigned outside a function have global scope. They are available inside and outside a function.

OUTPUT:

```
I love Algorithms
5
Have a nice day!!
```

Local variables, local scope

```
def f():
    """
     This function uses local
    variable ss and xx
    """
    # Local variables:
    ss = "I love Algorithms & CP"
    xx = 10
    return "Have a nice day!!"

print(f())
print(ss)
print(xx)
```

- Local variables are not available outside a function. They are created, used and deleted.

<u>OUTPUT</u>

```
NameError: name 'ss' is not defined
```

Local variables, local scope

```
def f():
    """
    This function uses local    variable
    ss and xx
    """
    # Local scope
    ss = "I love Algorithms and CP"
    xx = 10
    print(ss)
    print(xx)
    return "Have a nice day!!"

print(f())
```

Local variables are not available outside a function. They are created, used, and deleted once the function has been executed

<u>OUTPUT</u>

```
I love Algorithms and CP
10
Have a nice day!!
```

Global Statement

```
def f():
    """
    This function uses variable sss and
    xxx defined global within the
    function
    """
    # Global scope
    global sss, xxx
    sss = "I love Vegetarian Cooking"
    xxx = 1E-6

    return "Have a nice day!!"

print(f())
print(sss)
print(xxx)
```

It is possible to assign global variables within a function by using Python’s global statement

OUTPUT:

```
Have a nice day!!
I love Vegetarian Cooking
1e-06
```

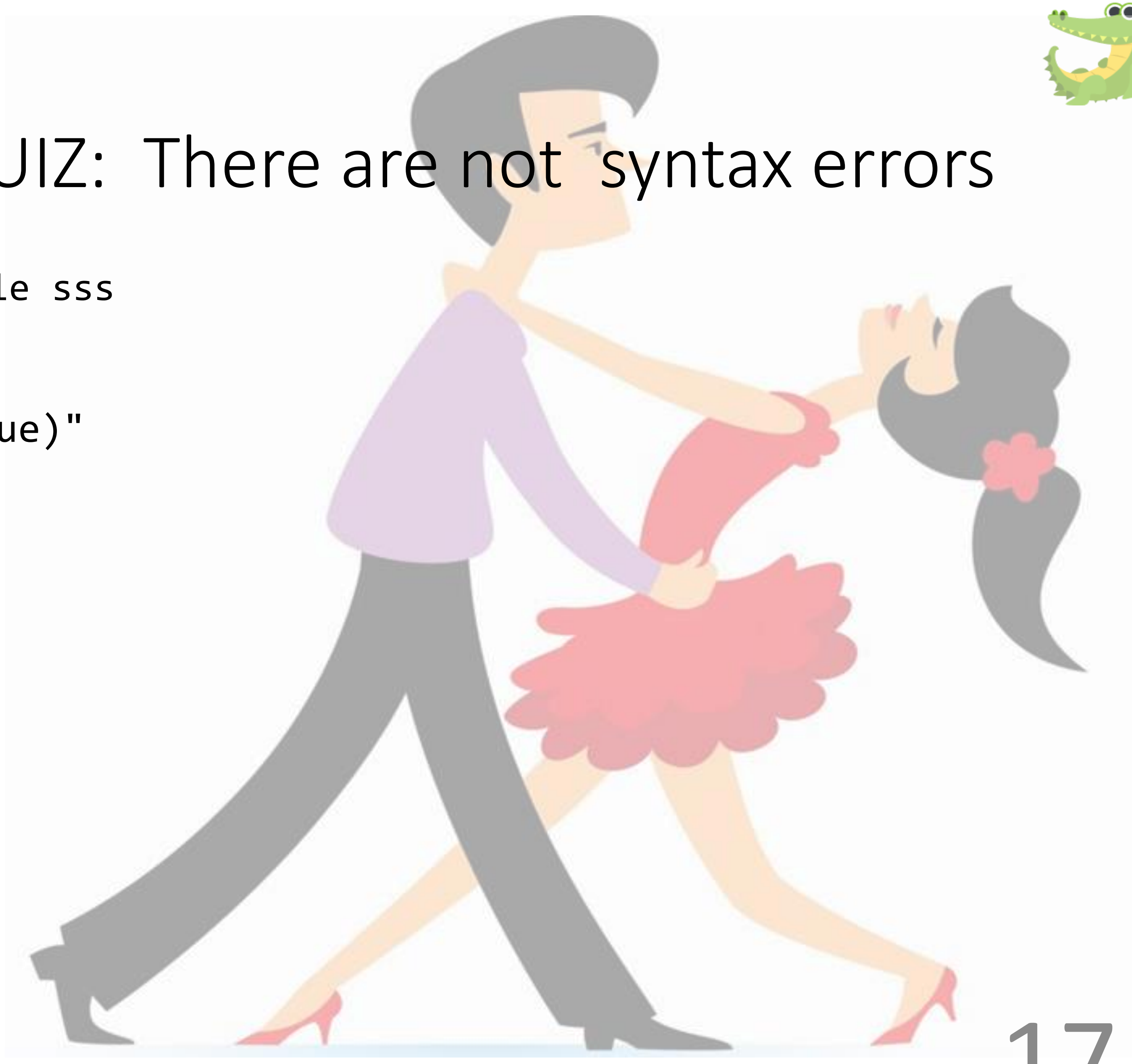
QUIZ: There are not syntax errors

```
# Global Scope
sss="I love Tacos"
xxx=1E6     # xxx=1000000

def f(sss,xxx):
     """ This function uses variable sss
    and xxx
     """
     sss += " con salsa (o merengue)"
     print(sss)
     xxx += 1E6
     print(xxx)

     return "Have a nice day!!"

print(f(sss,xxx))
print(sss)
print(xxx)
```



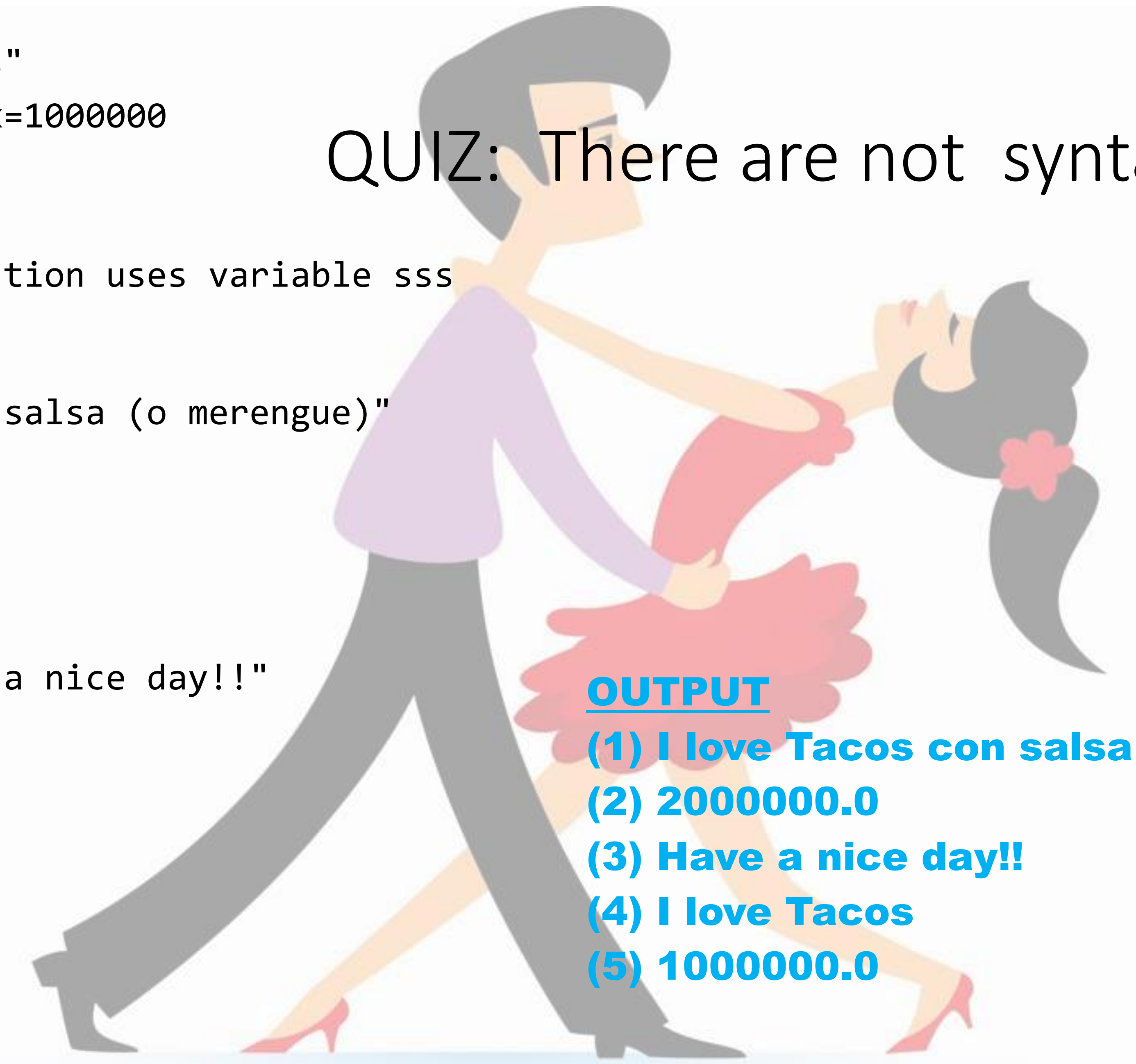
QUIZ: There are not syntax errors

```
# Global Scope
sss="I love Tacos"
xxx=1E6     # xxx=1000000

def f(sss,xxx):
    """ This function uses variable sss
and xxx
    """
    sss += " con salsa (o merengue)"
    print(sss)
    xxx += 1E6
    print(xxx)

    return "Have a nice day!!"

print(f(sss,xxx))
print(sss)
print(xxx)
```



OUTPUT

```
(1) I love Tacos con salsa (o merengue)
(2) 2000000.0
(3) Have a nice day!!
(4) I love Tacos
(5) 1000000.0
```

QUIZ

```
def high21(x):
    # Super…
    z=13+x
    return 5+z

def f(x,y):
    # More of a sum
    z = high21(x)
    return x+y+z

z=5
result = f(3,2)
print(result)
print(z)
```

```
def high21(x):
    # Super…
    z=13+x
    return 5+z

def f(x,y):
    # More of a sum
    z = high21(x)
    return x+y+z

z=5
result = f(3,2)
print(result)
print(z)
```

QUIZ

OUTPUT

```
26
5
```

Variable Scope: Local and Global Variables

- REFERENCE:
- https://www.digitalocean.com/community/tutorials/how-to-use-variables-in-python-3

Quiz-Exercises

1. Modify the function minimum to construct a function which finds the **maximum** value of a set of numbers stored in x, y, and z
2. Modify the function minimum2 to construct a function which finds the maximum of a set of numbers stored in array x
3. You can start solving SIMPLE FUNCTIONS LABORATORY Exercises in document: SIMPLE FUNCTION LABORATORY Exercises-KEY.docx