UTEC

CIENCIA DE LA COMPUTACIÓN

Funciones II

CS1100 - Introducción a Ciencia de la Computación
UTEC

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Determinar el alcance de una variable.

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Determinar el alcance de una variable.
- Crear y utilizar módulos en Python.

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Determinar el alcance de una variable.
- Crear y utilizar módulos en Python.
- Definir funciones utilizando parámetros por defecto.

Video Motivacional

Link

Alcance de una Variable

- Dependiendo de su alcance, existen dos tipos de variables: globales y locales.
- Variables globales: declaradas fuera de una función y utilizadas dentro de una función.

Alcance de una Variable

- Dependiendo de su alcance, existen dos tipos de variables: globales y locales.
- Variables globales: declaradas fuera de una función y utilizadas dentro de una función.

```
1       # Esta funcion utiliza la variable globales
2       def f():
3         print(s)
4       # Alcance global
5       s = "Que bonito es programar!"
6       f()
```

- Si se define una variable dentro una función con el mismo nombre que una variable global, ésta tomará el valor asignado dentro de la función.

Alcance de una Variable

- Dependiendo de su alcance, existen dos tipos de variables: globales y locales.
- Variables globales: declaradas fuera de una función y utilizadas dentro de una función.

```
1       # Esta funcion utiliza la variable globales
2       def f():
3         print(s)
4       # Alcance global
5       s = "Que bonito es programar!"
6       f()
```

- Si se define una variable dentro una función con el mismo nombre que una variable global, ésta tomará el valor asignado dentro de la función.

```
1     #La funcion f define una variable con el mismo nombre de una variable global (s)
2       def f():
3         s = "Electrónica también!."
4         print(s)
5       # Variable global
6       s = "CS es la voz!"
7       f()
8       print(s)
```

La asignación que se realiza dentro de la función, ¿afectará tambien la variable global?

Alcance de una Variable (2)

- ¿Qué crees que pasaría si se ejecuta el siguiente código?

```
1      def f():
2        print(s)
3        # ERROR????
4        s = "Electrónica también!"
5        print(s)
6      # Alcance global
7      s = "CS es la voz!"
8      f()
9      print(s)
```

Alcance de una Variable (3)

- En caso se requiera modificar el valor de una variable global dentro de una función se debe utilizar la palabra *global*.

```
1       # Esta función  modifica la variable global s
2       def f():
3         global s
4         print(s)
5         s = "Electrónica también!"
6         print(s)
7       # Alcance global
8       s = "CS es la voz!"
9       f()
10      print(s)
```

Módulos en Python

- Un módulo es un archivo que contiene definiciones e instrucciones Python.
- Puede definir funciones, clases y variables.
- Agrupar el código relacionado en diversos módulos hace el código más facil de entender y de usar.
- Ejemplo (calc.py):

```
1       # Un módulo simple, calc.py
2       def add(x, y):
3         return (x+y)
4
5       def subtract(x, y):
6         return (x-y)
```

Módulos en Python: *import*

- Para utilizar un archivo Python como un módulo, se debe utilizar la instrucción *import*.
- Cuando el intérprete encuentra una instrucción *import* importa el módulo si éste está presente en la ruta de búsqueda.
- La ruta de búsqueda es una lista de directorios en los cuales el intérprete busca los módulos a importar.
- Ejemplo:

```
1  # Un módulo simple, calc.py
2  def add(x, y):
3    return (x+y)
4
5  def subtract(x, y):
6    return (x-y)
```

```
1  # ejemplo.py, importando el modulo calc.py
2  import calc
3  print calc.add(10, 2)
```

Módulos en Python: *from*

- La instrucción *from* permite importar atributos específicos de un módulo.
- Ejemplo:

```
# importando sqrt() y factorial del módulo math
from math import sqrt, factorial
# Si simplemente se hubiera colocado "import math", sería
# necesario colocar math.sqrt(16) y math.factorial(6)
print sqrt(16)
print factorial(6)
```

- El ejemplo inicial, utilizando from:

```
# ejemplo.py, importando el modulo calc.py
from calc import add
print add(10, 2)
```

Parámetros por Defecto

- Python permite declarar parametros con valor por defecto. Si se invoca la función sin este parámetro, éste toma el valor por defecto que le fue asignado.
- El valor por defecto es asignado al parámetro haciendo uso del operador de asignación (=).
- Ejemplo:

```
1         def estudiante(nombre, apellido ='Rodriguez', standard ='Quinto'):
2           print(nombre, apellido, 'estudia en ', standard, 'Standard')
```

En este caso, la función estudiante tiene tres parámetros, de los cuales dos tienen valores por defecto asignados, por lo tanto, esta función tiene un parámetro obligatorio y dos opcionales.

Parámetros por Defecto(2)

- Al invocar funciones, es importante considerar que existen dos maneras de pasar los parámetros:
  - Parámetros posicionales (sin *keyword*).

```
def estudiante(nombre, apellido ='Mark', standard ='Quinto'):
  print(nombre, apellido, 'estudia en', standard, 'Standard')

# 1 parametro posicional
estudiante('John')
# 3 parametros posicionales
estudiante('John', 'Gates', 'Setimo')
# 2 parametros posicionales
estudiante('John', 'Gates')
estudiante('John', 'Setimo')
```

  - Parámetros con *keyword*.

```
def estudiante(nombre, apellido ='Mark', standard ='Quinto'):
  print(nombre, apellido, 'estudia en', standard, 'Standard')

# 1 parametro con keyword
estudiante(nombre ='John')
# 2 parametros con keyword
estudiante(nombre ='John', standard ='Seventh')
# 2 parametros con keyword keyword arguments
estudiante(apellido ='Gates', nombre ='John')
```

Parámetros por Defecto(3)

- Es importante siempre tener presente los siguientes puntos cuando se invoca a funciones:
  1. En caso se utilice *keyword* el orden de los parámetros no es importante.
  2. Únicamente debe haber un valor para cada parámetro.
  3. El nombre de los parámetros debe coincidir cuando se utilice *keyword*.
  4. En caso se invoque a la función sin *keyword*, el orden en que se pasen los parámetros es importante.

Ejercicios

1. Implemente las función `area_triangulo` de tal manera que el siguiente código funcione correctamente:

```
print(area_triangulo())
print(area_triangulo(2))
print(area_triangulo(2, 1))
#OUTPUT:
#1.0
#1.0
#1.0
```

2. Cree un módulo figura (figura.py), el cual implemente funciones para calcular el área de un círculo, un cuadrado y un triángulo. Posteriormente, invoque cada una de estas funciones desde un archivo main.py.
3. De las lineas enumeradas del 1 al 4, indique cuales de ellas generarían un error

```
def test(c):
  d = 3
  print(c)
  print(d)

a = 0
if a == 0:
  b = 1
test(7)

print(a) # (1)
print(b) # (2)
print(c) # (3)
print(d) # (4)
```

No olvidar..

- Dependiendo de su alcance, una variable puede ser local o global.
- Idealmente, debe evitarse utilizarse variables globales.
- Los parámetros por defecto permiten implementar código más flexible.
- Los módulos permiten tener código mas ordenado y reutilizable.