UNIVERSITY of HOUSTON | TECHNOLOGY

CIS 2348 University of Houston Information System Application Development

FALL 2021

CHAPTER 7

FUNCTIONS

FUNCTION DEFINITION

SYNTAX

```
DEF FUNCTION_NAME(PARAMETER1, PARAMETER2,..):
        ACTION CODE BLOCK
        RETURN RETURN_VALUE1, RETURN VALUE2, ..  #RETURN IS OPTIONAL

VALUE1, VALUE2,… = FUNCTION_NAME(ARGUMENT1, ARFGUMENT2,…)
```

WHEN ARE FUNCTIONS USEFUL

- WHEN YOU NEED EXECUTE THE SAME SET OF CODE MULTIPLE TIMES, BUT NOT SEQUENTIALLY
- WHEN YOU WANT TO MAKE IT VERY CLEAR IN YOUR CODE WHAT THE OPERATION IS AND WHAT ARE ITS INPUTS AND OUTPUTS
- WHEN YOU WANT IT TO BE AVAILABLE VIA IMPORT IN OTHER FILES

SIMPLE FUNCTION EXAMPLE

```
FROM MATH IMPORT SQRT
DEF GEOMETRIC_MEAN(A,B):
        GEOM_MEAN=SQRT(A*B)
        RETURN GEOM_MEAN
X=7
Y=9
PRINT('THE GEOMETRIC MEAN OF ',X,' AND ',Y,' IS ',GEOMETRIC_MEAN(X,Y))
```

FOR READABILITY WRITE CODE IN A MODULAR WAY

```
DEF SQUARE(I):
        RETURN (I**2)
DEF CUBE(I):
        RETURN (I**3)
LIST=RANGE(2,100)
LIST_SQUARES=[]
LIST_CUBES=[]
FOR I IN LIST:
        LIST_SQUARES.APPEND(SQUARE(I))
        LIST_CUBES.APPEND(CUBE(I))
```

MULTIPLE RETURN VALUES

```
FROM MATH IMPORT SQRT
DEF BOTH_MEANS(A,B):
        GEOM_MEAN=SQRT(A*B)
        ARITH_MEAN=(A+B)/2.0
        RETURN ARITH_MEAN,GEOM_MEAN
X=7
Y=9
A_MEAN,G_MEAN = BOTH_MEANS(X,Y)
PRINT('THE SUM OF THE GEOMETRIC AND ARITHMETIC MEANS OF ',X,' AND ',Y,' IS ',A_MEAN
+ G_MEAN)
```

WHEN TO USE MULTIPLE RETURNS

- WHEN YOU WANT TO MAKE IT VERY CLEAR WHICH INDIVIDUAL VARIABLES ARE SET BY THE FUNCTION
- WHEN THE NUMBER OF RETURNS IS SMALL
- WHEN THE NUMBER OF RETURN VALUES IS FIXED

PASSING ARGUMENTS BY KEYWORDS

ARGUMENTS CAN BE PASSED BY USING KEYWORDS, SO THAT THE ORDER IS INDEPENDENT OF ORDER IN THE FUNCTION DEFINITION

SYNTAX:

```
DEF FUNCTION_NAME(A,B):
        ACTION_CODE_BLOCK


FUNCTION_NAME(B=VAL1,A=VAL2)
```

USING KEYWORDS EXAMPLE

```
DEF COMPUTE_TOTAL_PRICE(PRICE_ORANGE,NUMBER_ORANGE,PRICE_APPLE,
NUMBER_APPLE):
        TOTAL=PRICE_ORANGE*NUMBER_ORANGE+PRICE_APPLE*NUMBER_APPLE
        RETURN TOTAL
TOTAL_PRICE=COMPUTE_TOTAL_PRICE(PRICE_APPLE=1.00,PRICE_ORANGE=0.50,NUMBER
_APPLE=5,NUMBER_ORANGE=8)
TOTAL_PRICE= COMPUTE_TOTAL_PRICE(0.50, 8, NUMBER_APPLE=5,PRICE_APPLE=1.0)
TOTAL_PRICE= COMPUTE_TOTAL_PRICE(0.50, 8, 1.0, 5)
```

DEFAULT VALUES OF PARAMETERS

- FUNCTION DEFINITIONS CAN HAVE DEFAULT VALUES OF PARAMETERS
- THESE VALUES WILL BE USED IF THE CORRESPONDING PARAMETERS ARE NOT PASSED IN
- THIS GIVES ADDITIONAL FLEXIBILITY TO THE FUNCTION

DEFAULTS PARAMETER EXAMPLE

```
DEF COMPUTE_TOTAL_PRICE(NUMBER_ORANGE, NUMBER_APPLE, PRICE_ORANGE=0.75,
PRICE_APPLE=1.25):
        TOTAL=PRICE_ORANGE*NUMBER_ORANGE+PRICE_APPLE*NUMBER_APPLE
        RETURN TOTAL
TOTAL_PRICE= COMPUTE_TOTAL_PRICE(10, 8, 1.0, 0.5) #SET BOTH PRICES
TOTAL_PRICE= COMPUTE_TOTAL_PRICE(10, 8)  #USE JUST NUMBER OF ITEMS
TOTAL_PRICE= COMPUTE_TOTAL_PRICE(10, 8, 1.0)  #SET PRICE OF ORANGES
```

WHAT IF I ONLY WANT TO SET PRICE OF APPLES?

PASSING PARAMETERS BY KEYWORDS AND DEFAULT VALUES

- THESE ARE OFTEN USED TOGETHER TO ALLOW SETTING THE VALUES FOR SOME OF THE PARAMETERS THAT HEAVE DEFAULTS

- KEEP IN MIND WHAT MAKES THE CODE MOST READABLE

- KEYWORDS MAKE THE ARGUMENTS MORE EXPLICIT, BUT FUNCTION CALLS MESSIER

OBJECTS CAN BE USED AS ARGUMENTS

LISTS, DICTIONARIES AND SETS MAY ALL BE USED AS PARAMETERS OF A FUNCTION

EXAMPLE

```
DEF SUM_OF_SQUARES(MY_OBJECT):
        SUM=0
        FOR I IN MY_OBJECT:
                SUM+=I**2
        RETURN SUM
MY_LIST=RANGE(4,99,7)
LIST_SUM=SUN_OF_SQUARES(MY_LIST)
MY_SET={9,45,39,786}
SET_SUM=SUM_OF_SQUARES(MY_SET)
```

DYNAMIC TYPING

- NO NEED TO DECLARE OBJECT TYPE IN THE FUNCTION PROTOTYPE
- THE INTERPRETER WILL PASS THE OBJECT AS NEEDED
- THE FUNCTION CODE NEEDS TO BE EITHER:
  - UNIVERSAL -- WORKS APPROPRIATELY FOR DIFFERENT TYPES
  - CONTAIN CONDITIONALS THAT SELECT THE CODE BASED ON ARGUMENT TYPE (NOT RECOMMENDED)
- IF THE TYPE IS NOT SUPPORTED IN THE FUNCTION WILL RETURN A RUN-TIME ERROR

FUNCTIONS WITH FLEXIBLE NUMBER OF PARAMETERS

- USE *ARGS TO DEFINE A FUNCTION WHERE THE NUMBER OF PARAMETERS IS FLEXIBLE
- WILL PUT THE CORRESPONDING PARAMETERS IN A LIST
- FUNCTION CAN PROCESS THE LIST
- ALLOWS FOR GREATER FLEXIBILITY AND CLEARER VIEW OF WHAT IS PASSED IN

USE OF *ARG EXAMPLE

```
DEF PRINT_MY_FRIEND(*ARGS):
        FOR NAME IN ARGS:
                PRINT(NAME,' IS MY FRIEND')
PRINT_MY_FRIEND("BOB", "ALICE")
PRINT_MY_FRIEND("ERIC")
PRINT_MY_FRIEND("BOB", "ALICE","JOHN")
```

CAN USE LIST INSTEAD OF *ARGS

```
DEF PRINT_MY_FRIEND(ARGS):
        FOR NAME IN ARGS:
                PRINT(NAME,' IS MY FRIEND')
MY_LIST=["BOB", "ALICE"]
PRINT_MY_FRIEND(MY_LIST)
```

WHY USE ONE VS THE OTHER?

USE OF **KWARGS

- ALLOWS PASSING IN OF ARBITRARY NUMBER OF ARGUMENTS BY KEYWORDS
- KEYWORDS AND THEIR VALUES ACCESSED BY KWARGS.ITEMS()

*ARGS AND **KWARGS CAN BE USED TOGETHER IN FUNCTION PROTOTYPE

*ARGS NEXT TO LAST, **KWARGS LAST

IN FUNCTION CALL, LISTED ARGUMENTS COME FIRST, ARBITRARY ARGUMENTS NEXT AND ARBITRARY KEYWORD ARGUMENTS LAST

EXAMPLE

```
DEF PRINT_LOCATION(DOCUMENT,*ARGS,**KWARGS):
        PRINT('THE MAIN DOCUMENTS IS', DOCUMENT)
        IF LEN(ARGS)>0:
                PRINT('THE ADDITION DOCUMENTS ARE:')
                FOR DOC IN ARGS:
                        PRINT(DOC)
        FOR LOC_TYPE, ADDRESS IN KWARGS.ITEMS():
                IF (LOC_TYPE=='FILE'):
                        PRINT('THE FILE NAME IS ', ADDRESS)
                ELSEIF (LOC_TYPE=='URL'):
                        PRINT('THE WEB LOCATION IS ', ADDRESS)
                ELSE:
                        PRINT('FIND THESE HERE ',LOC_TYPE,ADDRESS)
```

VARIABLE SCOPE

- VARIABLES CAN BE DEFINED GLOBALLY OR LOCALLY
- LOCAL VARIABLES CAN HAVE SAME NAMES AS GLOBAL VARIABLES
- LOCAL VARIABLES ONLY ACCESSIBLE FROM THE SCOPE OF THE FUNCTION
- GLOBAL VARIABLES AVAILABLE TO ALL FUNCTIONS
- FUNCTIONS CALLED FROM OTHER FUNCTIONS HAVE ACCESS TO THE CALLERS VARIABLES FOR READING
- CRITICAL TO UNDERSTAND THE SCOPE

LOCAL VARIABLES NOT AVAILABLE OUTSIDE(EXAMPLE)

```
DEF ADD_VARIABLES(A,B,C):
        SUM1=A+B
        SUM2=SUM1+C
        RETURN SUM2
SUM=ADD_VARIABLES(3,7,8)
PRINT(SUM, SUM1) –THIS IS PROBLEM! WOULD NEED TO RETURN IT TO ACCESS
```

GLOBAL VARIABLES IN FUNCTIONS

CAN BE READ.

```
DEF TOTAL_PRICE(NUM_ORANGES):
        RETURN NUM_ORANGE*PRICE_ORANGE
PRICE_ORANGE=0.80
PRINT (TOTAL_PRICE(6), PRICE_ORANGE)
```

HOW TO MODIFY GLOBALS IN FUNCTIONS

USE GLOBAL STATEMENT

```
DEF TOTAL_PRICE(NUM_ORANGES):
        GLOBAL PRICE_ORANGE
        IF PRICE_ORANGE<1.0:
                PRICE_ORANGE=1.0
        RETURN NUM_ORANGE*PRICE_ORANGE
PRICE_ORANGE=0.80
PRINT (TOTAL_PRICE(6), PRICE_ORANGE)
----   6 1.0
```

SOME GUIDANCE

- AVOID USING GLOBALS DIRECTLY IN FUNCTIONS
- IF YOU WANT TO ACCESS THEM – PASS THEM IN AS PARAMETERS
- IF YOU WANT TO MODIFY THEM – RETURN THEM AS ARGUMENTS
- SCOPE RESOLUTION IS A VERY COMMON SOURCE OF ERRORS
- RARE SITUATIONS REQUIRES GLOBAL VARIABLE ACCESS

EXAMPLE OF ABOVE

```
DEF TOTAL_PRICE(NUM_ORANGES, PRICE_ORANGE):
        IF PRICE_ORANGE<1.0:
                PRICE_ORANGE=1.0
        RETURN NUM_ORNGE*PRICE_ORANGE, PRICE_ORANGE
PRICE_ORANGE=0.80
TOTAL, PRICE_ORANGE =TOTAL_PRICE(6,PRICE_ORANGE)
PRINT (TOTAL_PRICE, PRICE_ORANGE)
```

FUNCTIONS ARE OBJECTS AND CAN BE ARGUMENTS

```
DEF ADD(A,B):
        RETURN A+B
DEF MULTIPLY(A,B):
        RETURN (A*B)
DEF OPERATE(OPERATION,A,B):
        RETURN  OPERATION(A,B)

OPERATE(ADD,8,9)
OPERATE(MULTIPLY,6,5)
```