Day - 10

**Functions with Outputs**

return statement,

* 1. **title()**

def title

***title() -> str*** Return a version of the string where each word is titlecased. More specifically, words ***start*** with ***uppercased*** ***characters*** and all ***remaining*** cased characters have ***lower*** ***case***.

**10.2 return statement**

The ***return*** keyword in Python exits a function and tells Python to run the rest of the main program. A return keyword can send a value back to the main program. While values may have been defined in a function, you can send them back to your main program and read them throughout your code.

* The concept is similar to C/C++/Java/C# and JavaScript. For example: **return** f"{fNm} {lNm}"

#*Name converter to Title- case*

**def** **TitLizer**(firstName, lastName):

    fNm = firstName**.title**()

    lNm = lastName**.title**()

**return** f"{fNm} {lNm}"

formattedName = **TitLizer**("KurE", "SaTler")

**print**(formattedName)

**print**(f"\n\t\t{**TitLizer**('PiLLI', 'soSo')}")

#*python retrn.py*

* Usage of Multiple-Return: For Exception handling ***return*** is mostly used.

#*Name converter to Title- case*

**def** **TitLizer**(firstName, lastName):

**if** (firstName **==** '') **or** (lastName **==** ''):

**return** f"No Empty Name"

    fNm = firstName**.title**()

    lNm = lastName**.title**()

**return** f"{fNm} {lNm}"

**print**(f"\n\t\t{**TitLizer**(input('Enter first-name : '), input('Enter last-name : '))}")

#*python multi\_return.py*

* Exercise 10.1: Revisit Leap-Year and modify it, also use return to calculate the days of month.

**def** **is\_Leap**(yer):

**if** yer%4 **==** 0:

**if** yer%100 **==** 0:

**if** yer%400 **==** 0 :

**print**("\t\t Is a Leapaa Year \n\n")

**return** "Leap"

**else**:

**print**("\t\t Oh no!! Not Leap Year \n\n")

**return** "NotLeap"

**else**:

**print**("\t\t Is a Leap Year \n\n")

**return** "Leap"

**else**:

**print**("\t\t Not a Leap Year \n\n")

**return** "NotLeap"

**def** **days\_in\_month**(yr, mNth):

    month\_days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

**if** **is\_Leap**(yr) **==** "Leap":

        month\_days[1] = 29

**return** f"\nEntered month {mNth} of year {yr} has {month\_days[mNth - 1]} days\n"

#*Do NOT change any of the code below*

year = **int**(input("\t\tEnter the year ? "))

month = **int**(input("Enter a month: "))

days = **days\_in\_month**(year, month)

**print**(days)

#*python leap\_retrn.py*

**10.3 Dockstrings**

It’s a some kind of POP-UP message in EDITOR to notify the user about the function – what kind of values it takes, what it returns etc. kind of information.

* *Dockstrings* appear right-after the function-declaration and "*indented*" inside **""" """**
* Also **""" """** can be used as multi-line string instead of using **" "**. As we used before to print some ASCII-art.
* It is used for DOCUMENTATION of our source-code.

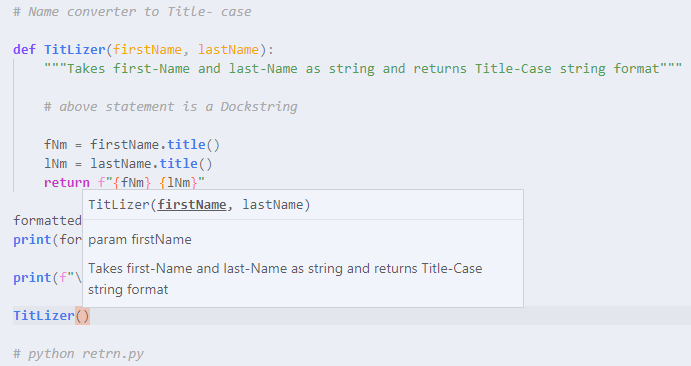
#*Name converter to Title- case*

**def** **TitLizer**(firstName, lastName):

    """Takes first-Name and last-Name as string and returns Title-Case string format"""

. . . .

. . . .



* Exercise 10.2: Calculator Project part-1.
* Notice:
* Notice the functions inside the dictionary " operations " there are no pranthesis "**()**" and no parameters are used. ***Only*** ***function*** ***names*** are used.

operations = {"+" : add, "-" : subtract, "\*" : multiply, "/" : devide}

* Also notice how selected function is retrieved from the dictionary and how the parameters are used.

result = operations[oPr](num1, num2)

#*Calculator*

num1 = **int**(input("Enter the first number : "))

num2 = **int**(input("Enter the second number : "))

#*Add*

**def** **add**(n1, n2):

**return** n1 + n2

#*subtract*

**def** **subtract**(n1, n2):

**return** n1 - n2

#*Multiply*

**def** **multiply**(n1, n2):

**return** n1 \* n2

#*Devide*

**def** **devide**(n1, n2):

**if**(n2**>**0):

**return** n1 / n2

**else**:

**return** "undefined"

operations = {"+" : add, "-" : subtract, "\*" : multiply, "/" : devide}

**for** op **in** operations:

**print**(op)

oPr = **input**("\n\t Pick an operation from above : ")

result = operations[oPr](num1, num2)

**print**(f"\n\t{num1} {oPr} {num2} = {result}\n")

#*python calculator\_pt\_1.py*

**10.4 print() vs return**

***return*** is more useful for its output value. This value can be used later in any kind of operations but we cannot do those things with ***print()***. As a simple word, ***return*** can do ***print()***'s and also many other things.

**10.5 Recursion**

A function calls itself inside its definition.

* Exercise 10.3: Calculator Project part-2. While loop and finishing touch. Recursive call.

#*Calculator*

logo = """

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  |

| | Pythonista   0. | |  .----------------.  .----------------.  .----------------.  .----------------.

| |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| | | .--------------. || .--------------. || .--------------. || .--------------. |

|  \_\_\_ \_\_\_ \_\_\_   \_\_\_  | | |     \_\_\_\_\_\_   | || |      \_\_      | || |   \_\_\_\_\_      | || |     \_\_\_\_\_\_   | |

| | 7 | 8 | 9 | | + | | | |   .' \_\_\_  |  | || |     /  \     | || |  |\_   \_|     | || |   .' \_\_\_  |  | |

| |\_\_\_|\_\_\_|\_\_\_| |\_\_\_| | | |  / .'   \\_|  | || |    / /\ \    | || |    | |       | || |  / .'   \\_|  | |

| | 4 | 5 | 6 | | - | | | |  | |         | || |   / \_\_\_\_ \   | || |    | |   \_   | || |  | |         | |

| |\_\_\_|\_\_\_|\_\_\_| |\_\_\_| | | |  \ `.\_\_\_.'\  | || | \_/ /    \ \\_ | || |   \_| |\_\_/ |  | || |  \ `.\_\_\_.'\  | |

| | 1 | 2 | 3 | | x | | | |   `.\_\_\_\_\_.'  | || ||\_\_\_\_|  |\_\_\_\_|| || |  |\_\_\_\_\_\_\_\_|  | || |   `.\_\_\_\_\_.'  | |

| |\_\_\_|\_\_\_|\_\_\_| |\_\_\_| | | |              | || |              | || |              | || |              | |

| | . | 0 | = | | / | | | '--------------' || '--------------' || '--------------' || '--------------' |

| |\_\_\_|\_\_\_|\_\_\_| |\_\_\_| |  '----------------'  '----------------'  '----------------'  '----------------'

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

"""

#*Add*

**def** **add**(n1, n2):

**return** n1 + n2

#*subtract*

**def** **subtract**(n1, n2):

**return** n1 - n2

#*Multiply*

**def** **multiply**(n1, n2):

**return** n1 \* n2

#*Devide*

**def** **devide**(n1, n2):

**if**(n2**>**0):

**return** n1 / n2

**else**:

**return** "undefined"

operations = {"+" : add, "-" : subtract, "\*" : multiply, "/" : devide}

**def** **calculator**():

**print**(logo)

    num1 = **float**(input("Enter the first number : "))

    num2 = **float**(input("Enter the second number : "))

    eXit = **False**

**while** **not** eXit:

**for** op **in** operations:

**print**(op)

        oPr = **input**("\n\t Pick an operation from above : ")

        result = operations[oPr](num1, num2)

**print**(f"\n\t{num1} {oPr} {num2} = {result}\n")

        aSk = **input**(f"""Type \'ex\' to end or \'bg\' to satrt again.

                    Press enter/other-key to cintinue with result {result}""")

**if** aSk **==** "ex":

            eXit = **True**

**elif** aSk **==** "bg":

**calculator**()

            eXit = **True**

**else**:

            num2 = **float**(input("Enter another number : "))

            num1 = result

**calculator**()

#*python calculator\_pt\_1.py*

|  |  |
| --- | --- |
| * Notice: Right-sided code is used after recursion. The reason is if loop ends in recursive call the loop of outer-call still continues. | **elif** aSk **==** "bg":  **calculator**()              eXit = **True** |