PRG1

W
E
E
K

12

NGEE ANN
SCHOOL OF INFOCOMM TECHNOLOGY

Functions (Part 2)

Programming I (PRG1)

Diploma in Information Technology

Diploma in Financial Informatics

Diploma in Cybersecurity & Digital Forensics

Common ICT Programme

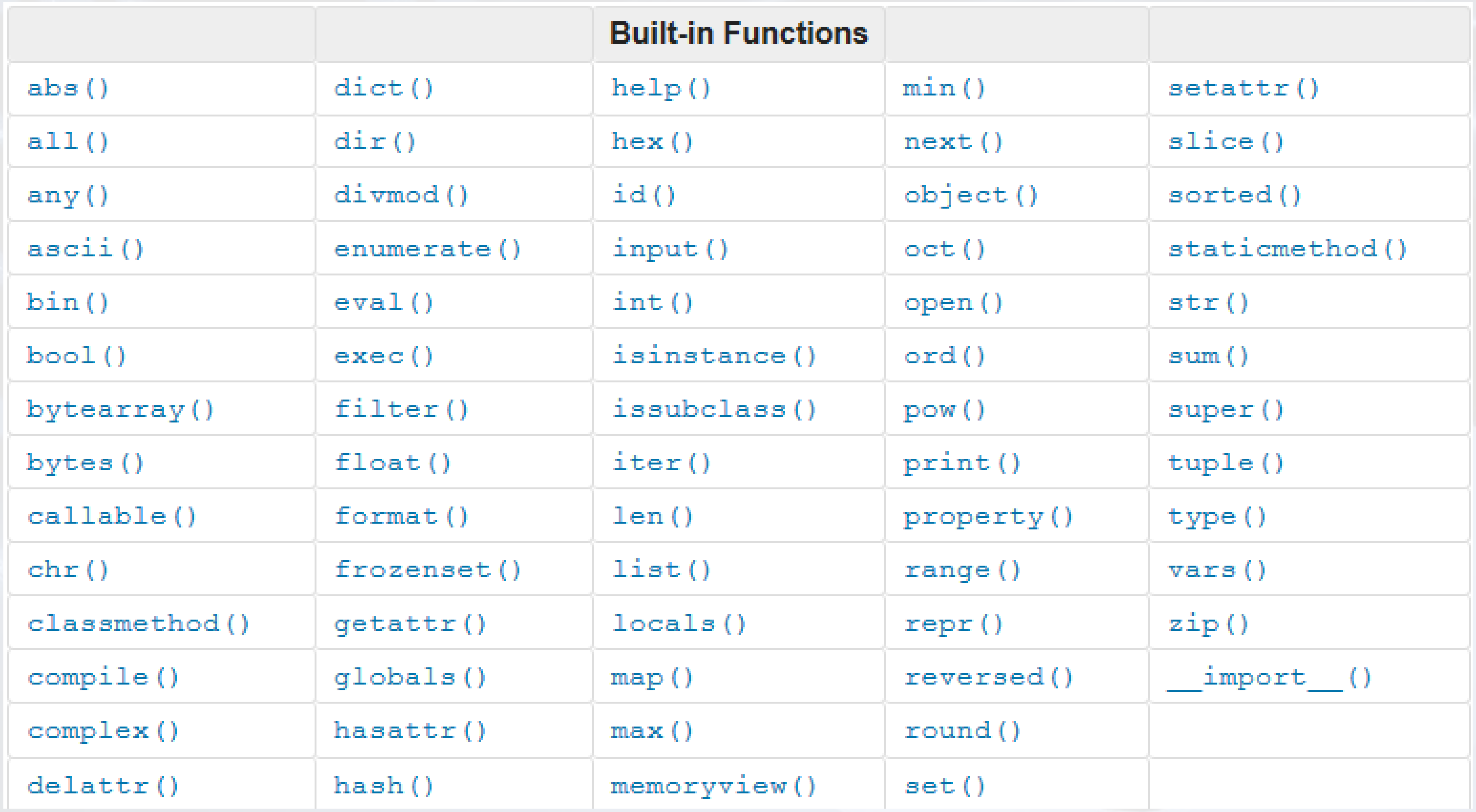
Year 1 (2019/20), Semester 1

Objectives

At the end of this lecture, you will ….

1. Built-in/pre-defined functions
2. Write function with > 1 return statement
3. Use function(s) in a loop
4. Scope

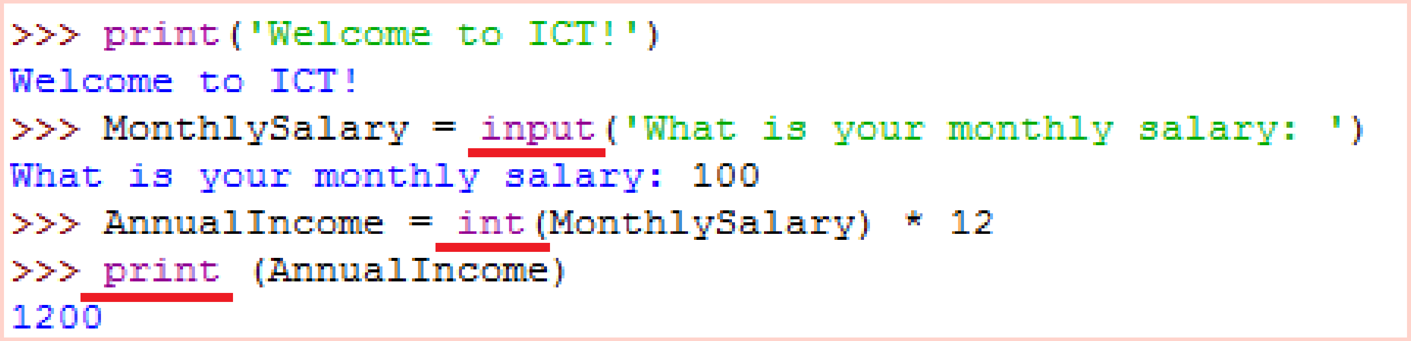
Recall Built-in functions

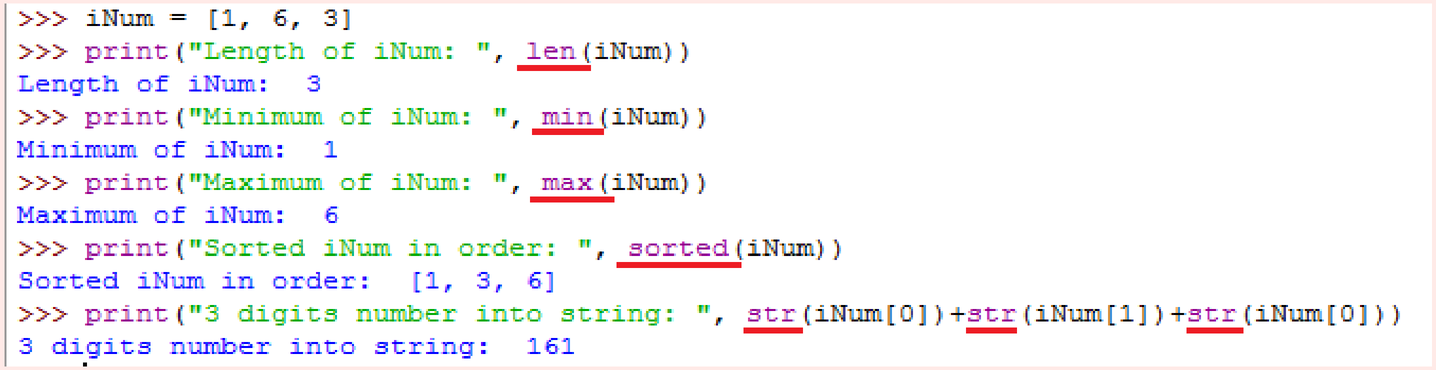


		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Commonly used built-in functions

Function	Usage

print()
input()
int()
	

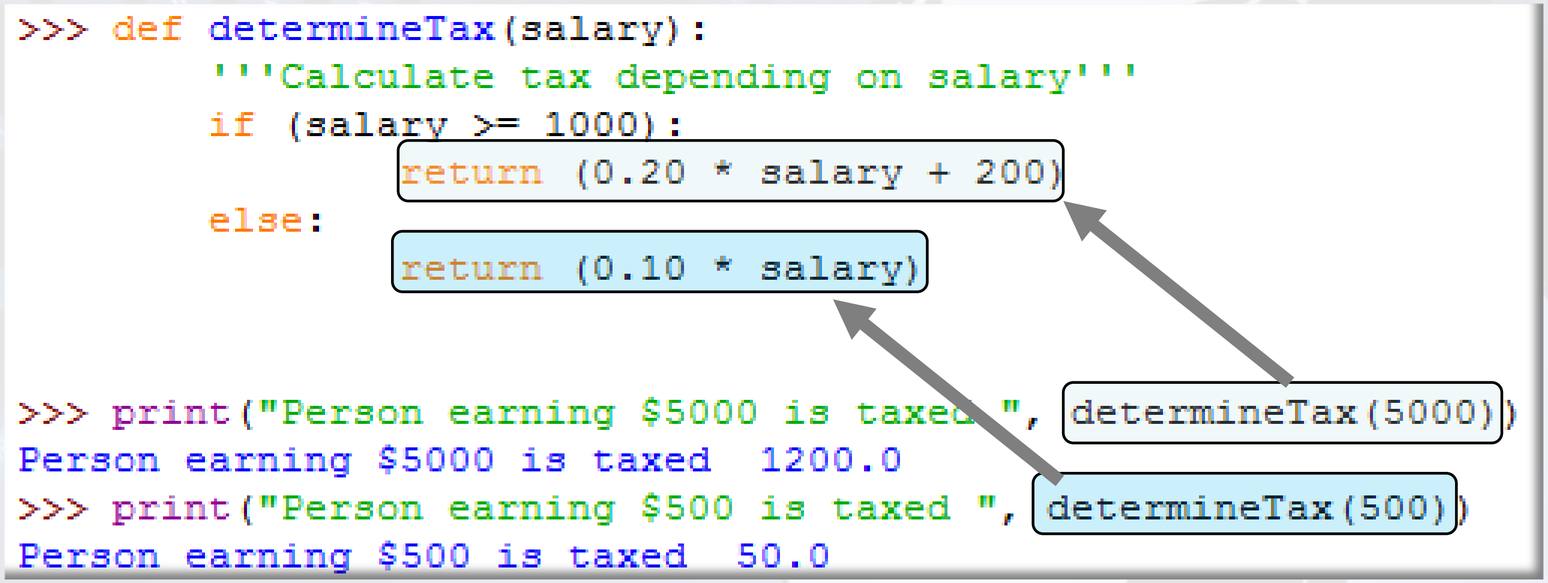
len()
min()
max()
sorted()
str()
	

```
>>> print('Welcome to ICT!')
Welcome to ICT!
>>> MonthlySalary = input('What is your monthly salary: ')
What is your monthly salary: 100
>>> AnnualIncome = int(MonthlySalary) * 12
>>> print (AnnualIncome)
1200
```

```
>>> iNum = [1, 6, 3]
>>> print("Length of iNum: ", len(iNum))
Length of iNum:  3
>>> print("Minimum of iNum: ", min(iNum))
Minimum of iNum:  1
>>> print("Maximum of iNum: ", max(iNum))
Maximum of iNum:  6
>>> print("Sorted iNum in order: ", sorted(iNum))
Sorted iNum in order:  [1, 3, 6]
>>> print("3 digits number into string: ", str(iNum[0])+str(iNum[1])+str(iNum[0]))
3 digits number into string:  161
```

Writing function with > 1 return statement

- A function may have more than one return statement



```
>>> def determineTax(salary):
        '''Calculate tax depending on salary'''
        if (salary >= 1000):
                return (0.20 * salary + 200)
        else:
                return (0.10 * salary)


>>> print("Person earning $5000 is taxed ", determineTax(5000))
Person earning $5000 is taxed  1200.0
>>> print("Person earning $500 is taxed ", determineTax(500))
Person earning $500 is taxed  50.0
```

- The function call results in different return value based on the specified condition
- At any time, only one value will be returned to the calling function

Using functions in a loop

- **Functions can be used in repetition loop**

  Consider the following:-

Computer paper usage from printers is charged at the following rates:

First 100 pages	: 3 cents a page
Next 200 pages	: 2 cents a page
Over 300 pages	: 1 cent a page

Determines the charge (inclusive of 7% GST) for printing of paper in stacks of 50 pages from 0 to 500 pages.

Using functions in a loop

- Define the following functions in the program:

  calculateCharge()
  - takes in the number of pages printed and return the corresponding charge

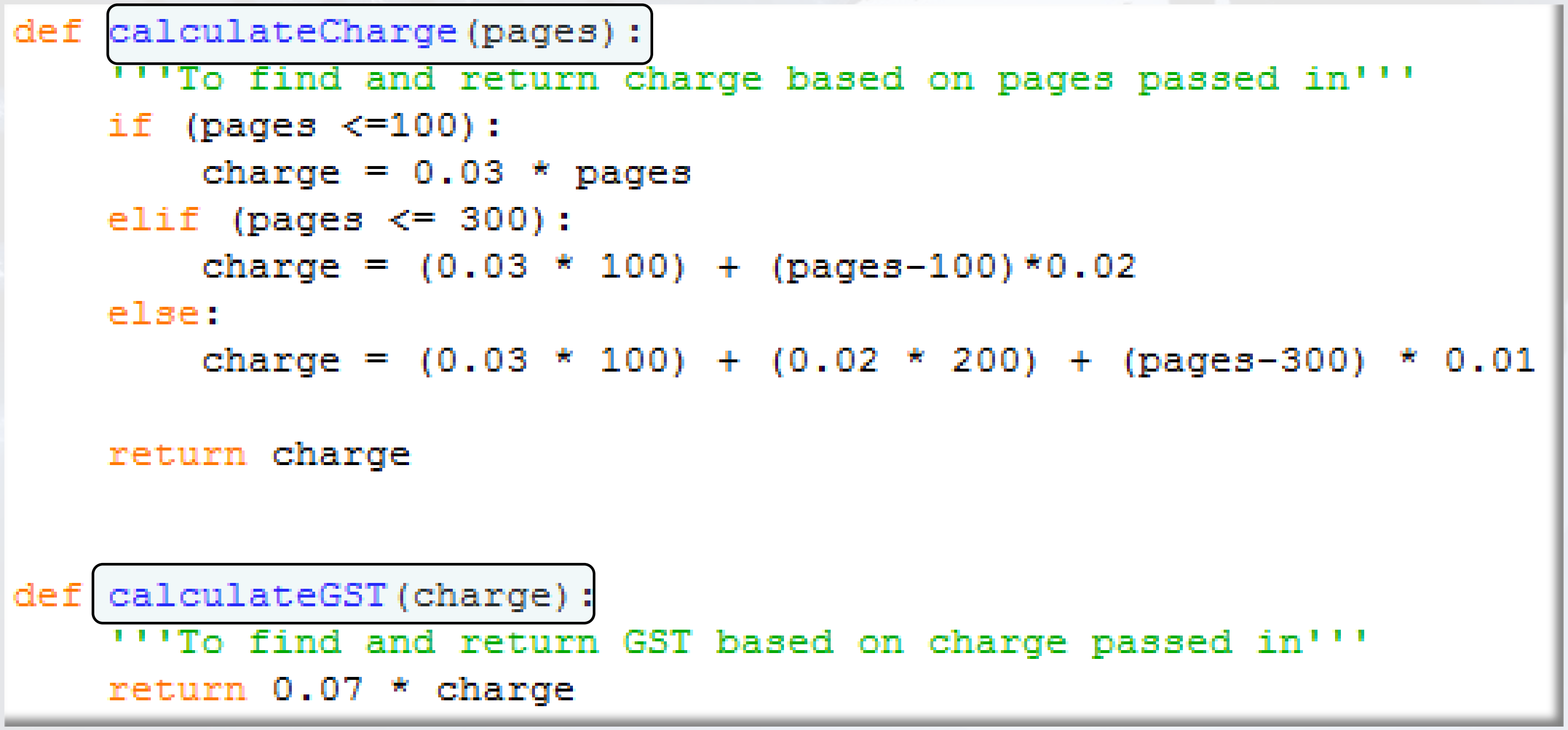
  calculateGST()
  - takes in the amount and return the corresponding GST charged

  The expected output is as follows:

```
Pages    Charge     Charge (include GST)
0        $0.0       $0.0
50       $1.5       $1.605
100      $3.0       $3.21
:        :          :
500      $9.0       $9.63
```

Using function in a loop

Step1: Define the functions involved



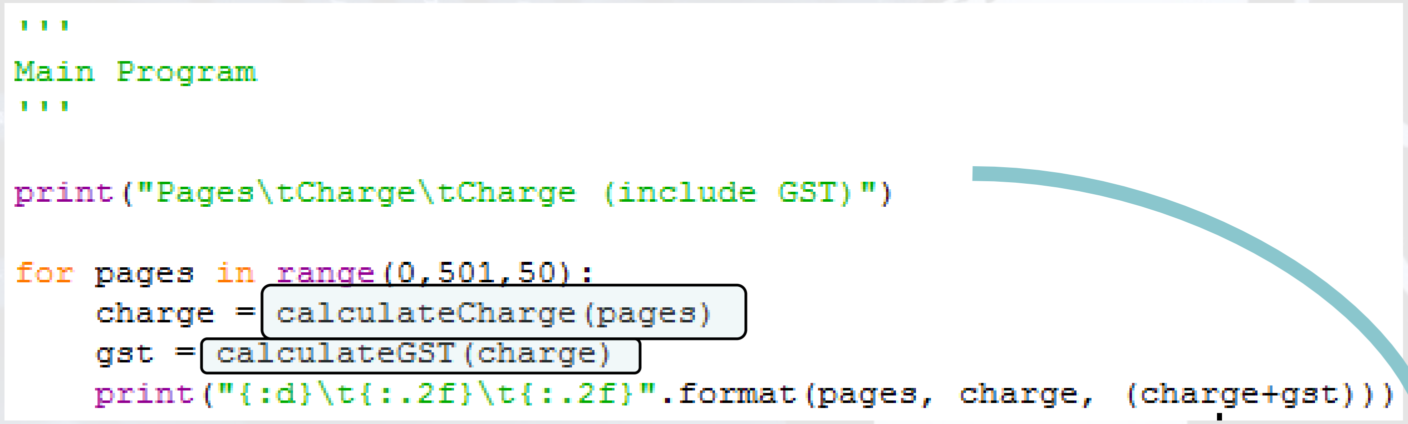
```
def calculateCharge(pages):
    '''To find and return charge based on pages passed in'''
    if (pages <=100):
        charge = 0.03 * pages
    elif (pages <= 300):
        charge = (0.03 * 100) + (pages-100)*0.02
    else:
        charge = (0.03 * 100) + (0.02 * 200) + (pages-300) * 0.01

    return charge


def calculateGST(charge):
    '''To find and return GST based on charge passed in'''
    return 0.07 * charge
```

Using function in a loop

Step 2: Make the function call in repetition loop

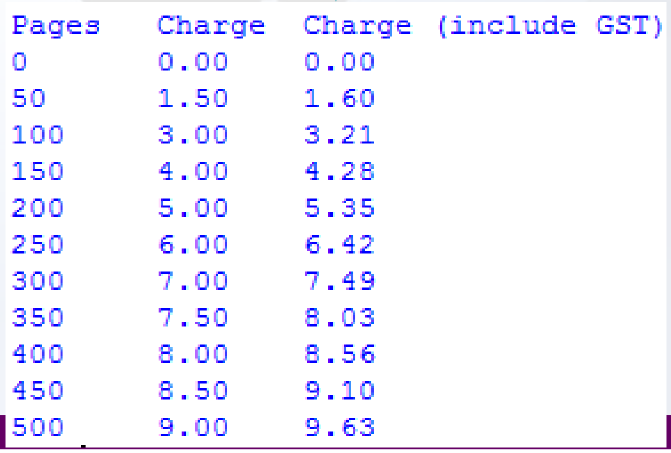


```
'''
Main Program
'''

print("Pages\tCharge\tCharge (include GST)")

for pages in range(0,501,50):
    charge = calculateCharge(pages)
    gst = calculateGST(charge)
    print("{:d}\t{:.2f}\t{:.2f}".format(pages, charge, (charge+gst)))
```

For every loop, the functions calculateCharge() and calculateGST() will be called with the respective parameters value



```
Pages    Charge    Charge (include GST)
0        0.00      0.00
50       1.50      1.60
100      3.00      3.21
150      4.00      4.28
200      5.00      5.35
250      6.00      6.42
300      7.00      7.49
350      7.50      8.03
400      8.00      8.56
450      8.50      9.10
500      9.00      9.63
```

Scope

Consider this example:

```
def f(x):
    x = x + 1
    print(x) #prints 6


x = 5
f(x)
print(x)#prints 5
```

```
Output: 6
        5
```

Why is this not 6?

Scope

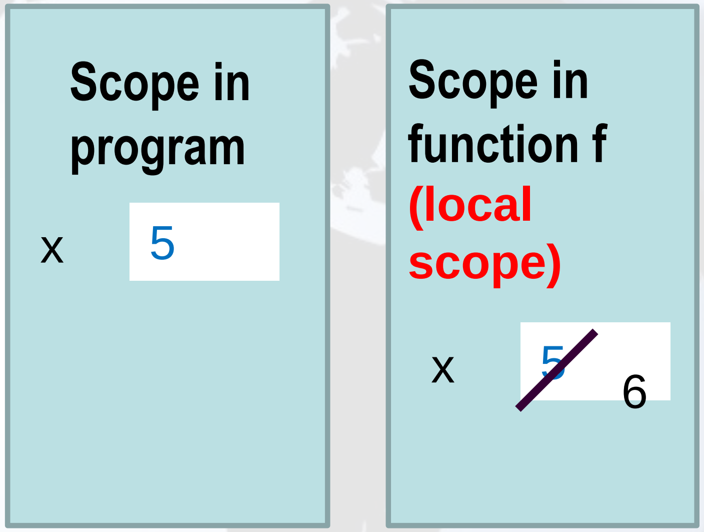
- The two x variables are referring to different data objects in the memory.

```
def f(x): #local scope
    x = x + 1
    print(x)#prints 6


x = 5 #program scope
f(x)
print(x)#prints 5
```

```
Output:  6
         5
```

Why is this not 6?



- And they have different scope: scope of one x is the program, while for the other x, it is the function

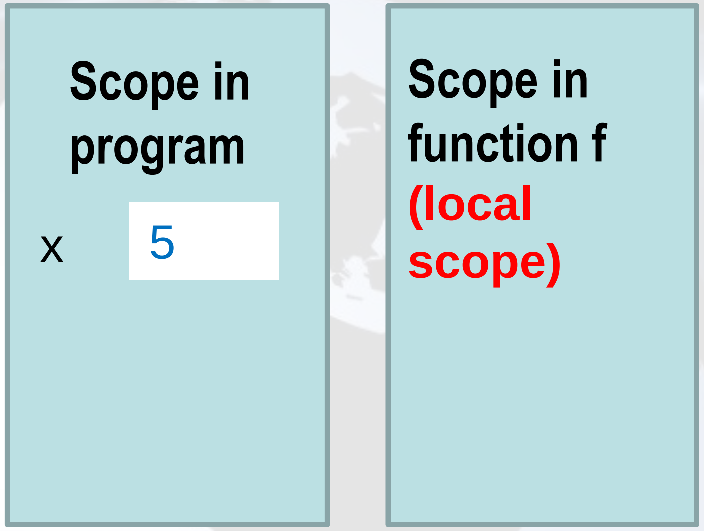
Scope

Consider this example now :

```
def f( ):
    x = x + 1
    print(x)

x = 5
f( )
print(x)
```

This line is expecting an x variable of local scope, with value defined beforehand!



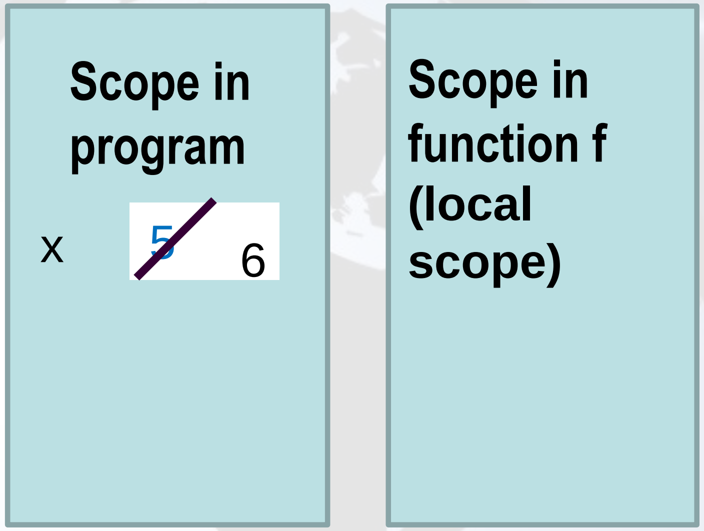
```
UnboundLocalError: local
variable 'x' referenced before
assignment
```

Scope

- *Scope* of a variable is the portion of the program that the data object can be referred to by its name.
- We say that the data object is ‘in scope’ for that portion of the program.
- Scope of a variable is usually the block of the program/function in which it is defined.
- i.e. the scope is usually from the point the variable is defined till the end of the block

Scope

So how do we make the x outside the function to be updated with the new value?



<u>Solution 1:</u>

```
def f( ):
    global x
    x = x + 1
    print(x)
x = 5
f( )
print(x)
```

Indicating that x with the global scope will be used.

```
Output:  6
         6
```

However, this technique is often frowned upon in programming! It reduces code maintainability as knowing where and how they were defined, especially in large programs, is difficult. . Using global variables also discourages encapsulation.

Scope

So how do we make the x outside the function to be updated with the new value?

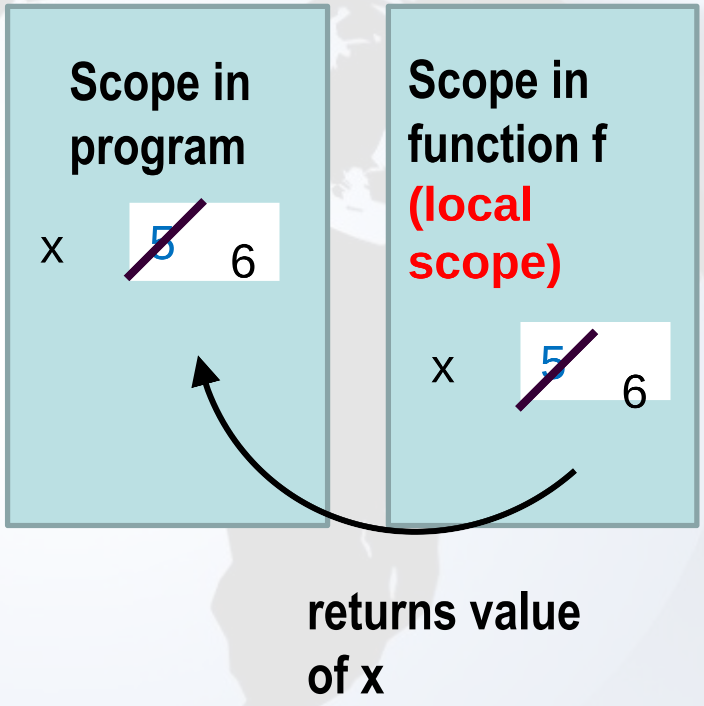
Solution 2:

```
def f(x):
    x = x + 1
    return x

x = 5
x = f( )
print(x)
```

Indicating that x with the global scope will be used.

```
Output: 6
        6
```



Activity 1: Obtain grade for student’s marks

- Write a function obtain_grade () that receives a float parameter as a student’s mark and returns the grade according to the following criteria:

Average	Grade
84.5 – 100	A+
79.5 – 84.5 exclusive	A
74.5 – 79.5 exclusive	B+
69.5 – 74.5 exclusive	B
64.5 – 69.5 exclusive	C+
59.5 – 64.5 exclusive	C
54.5 – 59.5 exclusive	D+
49.5 – 54.5 exclusive	D
Below 49.5	F

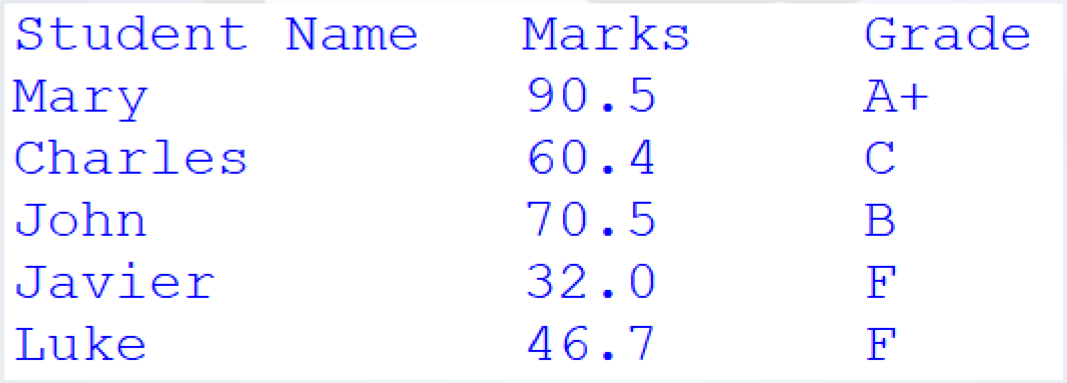
Activity 1: Obtain grade for student's marks

- Incorporate the function in a program that processes a list of students' averages and display the result of the averages in a tabular format.

- The list is as follows:

```
student_marks = [['Mary', 90.5], ['Charles', 60.4],
['John', 70.5], ['Javier', 32.0], ['Luke', 46.7]]
```

- Expected output:



```
Student Name     Marks       Grade
Mary             90.5        A+
Charles          60.4        C
John             70.5        B
Javier           32.0        F
Luke             46.7        F
```

Activity 2: Temperature Conversion

- Write a function fahr_to_cel( ) that returns the Celsius equivalent of a Fahrenheit temperature.

```
Celsius = 5.0/ 9.0 x (Fahrenheit – 32)
```

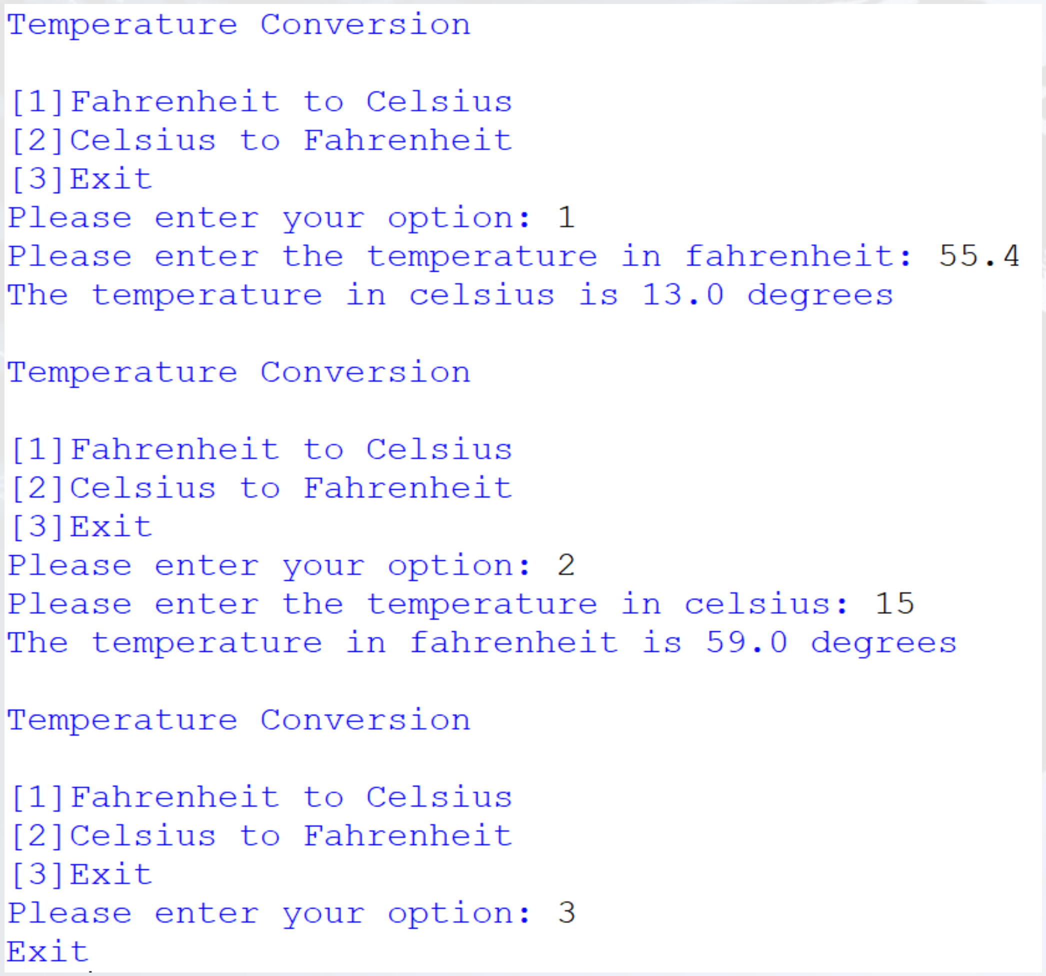
- Write another function cel_to_fahr( ) that returns the Fahrenheit equivalent of a Celsius temperature.

```
Fahrenheit = 9.0/ 5.0 x Celsius + 32
```

- Using the functions above, write a program that allows the user to choose between two menu options to do either of the two conversions and display the result.

Activity 2: Temperature Conversion

- **Here's a sample run of the program:**



```
Temperature Conversion

[1]Fahrenheit to Celsius
[2]Celsius to Fahrenheit
[3]Exit
Please enter your option: 1
Please enter the temperature in fahrenheit: 55.4
The temperature in celsius is 13.0 degrees

Temperature Conversion

[1]Fahrenheit to Celsius
[2]Celsius to Fahrenheit
[3]Exit
Please enter your option: 2
Please enter the temperature in celsius: 15
The temperature in fahrenheit is 59.0 degrees

Temperature Conversion

[1]Fahrenheit to Celsius
[2]Celsius to Fahrenheit
[3]Exit
Please enter your option: 3
Exit
```

Summary

- Recommended to use built-in/ pre-defined functions that are ready to use
- Function can be called in loops and may contain more than 1 return value
- It is important to understand scope of variables in order to know their access and visibility.