SCPY204: Computer programming for Physicists

Basic things to remember for C and Python

1. Data types and variable

1.0 Syntax

Syntax	C	Python
End a statement	;	; or not required
Comment (single line)	//	#
Comment (multiple lines)	/* … */	“ ” ” … “ ” ”
Code block	{ }	code indentation

1.1 Basic data types

- There are basic data types we are dealing with: integer, floating and string (or character). You need to know how to declare, define and assign a value to the variable of these data types.
- There are also secondary (derived) data types in each language:
  C: array, pointer and class etc.,
  Python: list, tuple and dictionary etc.

1.2 Memory

- Programming languages (C and Python) may have different scheme how a variable refers to a memory location.

2. Operator

2.1 Arithmetic operator

Operation	C	Python
Addition	+	+
Subtraction	-	-
Multiplication	*	*
Exponentiation	**	**
Division	/	/
Floored division	not available	//
Modulus	%	%

Note: Always parenthesize complex arithmetic operations.

2.2 Relational operators

Relation	C	Python
Greater than	>	>
Less than	<	<

Greater than or equal to	>=	>=
Is less than or equal to	>=	>=
Is equal to	==	==
Is not equal to	!=	!=

3. Control structures

3.1 if statements

IF block	C	Python

If	`if (x==y) {`
	`  printf("Equal\n");`
	`}`
		`if x==y:`
		`  print("Equal")`

If … else …	`if (x==y) {`
	`  printf("Equal\n");`
	`} else {`
	`  printf("Not equal\n");`
	`}`
		`if x==y:`
		`  print("Equal")`
		`else:`
		`  print("Not equal")`

If … else if .. else	`if (x==y) {`
	`  printf("Equal\n");`
	`} else if (x>y) {`
	`  printf("x greater\n");`
	`} else {`
	`  printf("y greater\n");`
	`}`
		`if x==y:`
		`  print("Equal")`
		`elif x>y:`
		`  print("x greater")`
		`else:`
		`  print("y greater")`

3.2 Iteration (loop)

Loop	C	Python

for	`for (ii=0;ii<10;ii++) {`
	`  printf("%d\n",ii);`
	`}`
		`for ii in range(10):`
		`  print(ii)`

while	`count = 0;`
	`while (count<10) {`
	`  printf("%d\n",count);`
	`  count = count+1;`
	`}`
		`count = 0`
		`while count<10:`
		`  print(count)`
		`  count = count+1`

Note: `break` can be used to stop and continue program flow outside the current loop.

4. Function and recursion

Function	C	Python

Void	`void print_me(int num){`
	`  printf("%d",num);`
	`  return;`
	`}`
		`def print_me(num):`
		`  print(num)`
		`  return`

Return a value	`int adding(int x,int y){`
	`  return x + y;`
	`}`
		`def adding(x,y):`
		`  return x+y`

Recursive	`int factorial(int n){`
	`  if (i<=1){`
	`  return 1;`
	`} else {`
	`  return n*factorial(n-1)`
	`}`
		`def factorial(n):`
		`  if i<=1:`
		`  return 1`
		`  else:`
		`  return n*factorial(n-1)`
		`}`

5. Collection of data

Sequence	C	Python
1-D	`int a = {1,2,3,4,5};` `printf("%d\n",a[0])`	`a = [1,2,3,4,5];` `print(a[0])`

2-D	`int b = {{1,2,3,4},`
	`{6,7,8,9}};`
	`printf("%d\n",a[0][0])`
		`b = [[1,2,3,4],`
		`[5,6,7,8]]`
		`print(a[0][0])`

6. Modules

C	Python

`#include<stdio.h>`
`#include<math.h>`
 	
`void main() {`
`  float x;`
`  x = sqrt(47.5);`
`}`
	`import math`
	`x = math.sqrt(47.5)`

7. Input and output

I/O	C	Python
Screen output (simple)	`Not available`	`x = 10` `print("x = ",x)`
Screen output (formatted)	`int x = 10;` `printf("x = %d\n",x)`	`x = 10` `print("x = %d" % x)`
Output format	`%[flags][width][.precision]type`	

Escape string	`\n (newline, line feed)`
	`\t (tab)`
	`\v (vertical tab)`
	`\f (new page)`
	`\b (backspace)`	