2020

Free Python Course

O. Elijah

onlinepythonclass@gmail.com

O. ELIJAH (MSc, Ph.D)

1 INTRODUCTION

Welcome to chapter one the free python course. In this chapter, we hope to achieve the following objectives.

1. Learn how to use the spyder integrated development environment (IDE)
2. Understand the simple python rules
3. Know what variables are and how to create and use variables
4. Learn some of the data types in python.

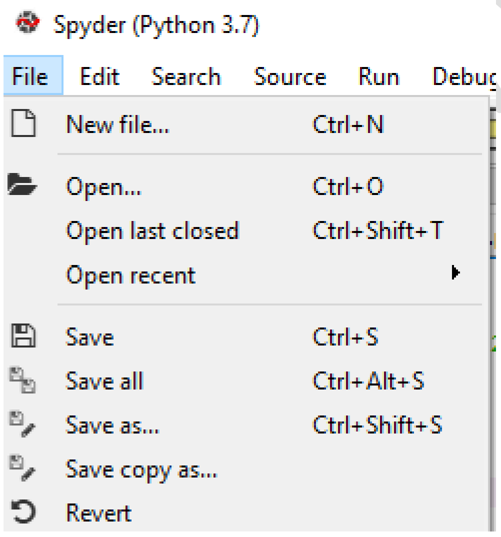
First, lets us have a look in how the computer works.

1.1 Using Spyder to practice the codes

Kindly click on the link below on how to use Spyder Software. Or view download the video from your week 1 folder.

https://www.youtube.com/watch?v=a1P_9fGrfnU

You can create a new file in the Spyder editor go to **file**-> **New file**



Save it as chapter1. To save go to **file**->**save**

You can type your codes in the Spyder editor and click on **Run** or the green arrow . The result will be displayed in the console.

Or you can practice your codes using the console. However, note that the codes you practiced using the console will be deleted when you close your spyder.

1.2 Comment

Comments are used to document your codes. It helps you and other users to read and understand your codes. There are two ways to comment in python: one is single line comments and multiply line comments.

Single line comment

Starts with the hash character, #

Example 1.

# This is my first program in FPC!

Multiple line comment

Multiple line comment is created using the following characters """ at the beginning of the line and """ at the end of the last line.

Example 2.

"""

This is my first program in FPC!

I hope to be a profession python programmer by June 2020

I will work hard and smart to complete the course

"""

Freeze your codes

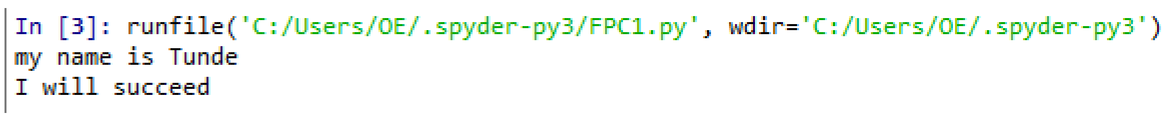
You can also comment part of your codes if you do not wish to execute them by using the # or the """ codes """

Example 3

```
"""                                  |
Created on Tue Jan 14 11:12:20 2020  |
                                     |Example of multiple line comment
@author: OE                          |
"""

print("my name is Tunde")
#print ("i want to be a programmer")   | example of commenting part of a code
print("I will succeed")
```

If you run the codes in Example 3, check the following output will be shown on the console. Notice the " I want to be a programmer" is not printed.



```
In [3]: runfile('C:/Users/OE/.spyder-py3/FPC1.py', wdir='C:/Users/OE/.spyder-py3')
my name is Tunde
I will succeed
```

1.3 Variables

Variable are like containers that stores your data.

Variable names can take many forms, although they can only contain numbers, letters (both upper and lower), and underscores (_).

Rules for creating a variable

a. Always use meaningful names for your variables,
b. start with a character or underscore
c. You cannot start with a digit
d. Variables are case sensitive. Example **Variable** and **variable** are not the same
e. Keywords cannot be used. Keywords are reserved names in Python. Examples of keywords are https://www.programiz.com/python-programming/keyword-list

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Assigning data to a variable

Example

A = 4

The computer takes the data 4 and stores in a variable called A. Here are other examples you can practice.

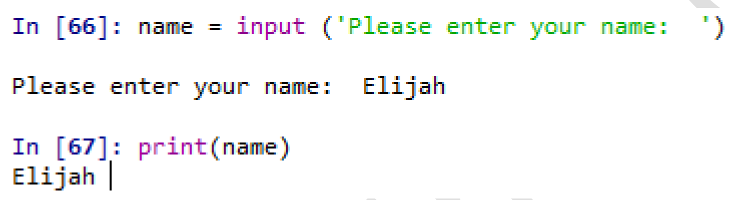
1. Assigning data to a variable or initialize your variables
   a. amount = 20
   b. name = "peter"
   c. price = 45.40
   d. print (amount, name, price)
2. Multiple variables can be assigned on the same line using commas
   a. x, y, z = 1, 3.1415, "peter"
3. age = 25

***Note the print function is used to display the data in a variable on the screen**

Take input from the keyboard and assigning to a variable

The function input is used here to get the input from your keyboard and the data is stored in the variable. Examples

Example 1 write the code to ask a user to enter his/her name. In programming this is called prompt

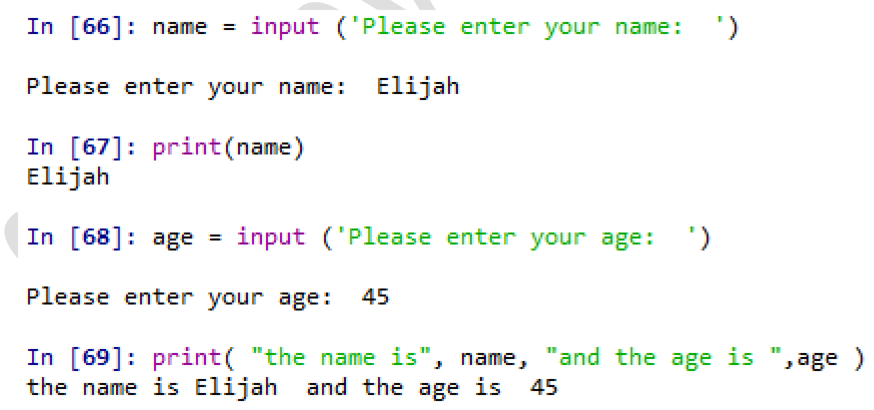


```
In [66]: name = input ('Please enter your name:  ')

Please enter your name:  Elijah

In [67]: print(name)
Elijah
```

Example 2: Ask the user to enter his name and age



```
In [66]: name = input ('Please enter your name:  ')

Please enter your name:  Elijah

In [67]: print(name)
Elijah

In [68]: age = input ('Please enter your age:  ')

Please enter your age:  45

In [69]: print( "the name is", name, "and the age is ",age )
the name is Elijah  and the age is  45
```

1.4 Data types

There are different data types in python. They are **numeric**, **boolean**, **sequence, set** and **dictionary**.

Numeric

Numeric data types consist of either integers, floats or complex numbers.

1. **Float** - any number that has a decimal point is considered a float
   X = 1.0
2. **Complex** - is a combination of real and imaginary numbers.

   X = 2 + 3j

3. **Integers** - An integer is a whole number (not a fractional number) that can be positive, negative, or zero

   X = 1 or X = -4 or X = 0

Note if you want to check the data type you can use the function **type().** In week 6 we shall learn what function is.

```
x = 1

type(x)
Out: int
Y = 34.60

type(Y)
Out: float

name = "steven"

type(name)
Out: str
```

4. **Boolean** - represent True and False or 1 and 0

   x = True or x = False

Sequence

Under sequence we have Strings, List and Tuple

1. **Strings** (str) – used for formatting outputs or dealing with data files (delimited using double quote " " or single quote ' '). Examples

```
 x = 'abc'

y = '''A quotation! '''

print(y)

Out: "A quotation! "
```

2. **List** - A list is a collection of other objects - floats, integers, complex numbers, strings or even other lists. Basic lists are constructed using square braces, [ ], and values are separated using commas. Examples

x = []

b = [1,2,3,4,5,6,7,8,9,10]

There are different kinds of list which are 1-dimensional and 2-dimensional list. They also known as 1-dimensional array and 2-dimensional arrays.

1-Dimensional list

Example of 1-dimensional list is

b = [1, 2, 3, 4, 5]

1-D array

1	2	3	4	5

2-Dimensional list

x = [[1,2,3,4,5],[4,0,30,5,6]]

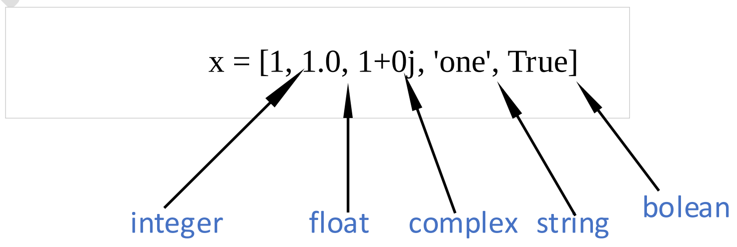
2-D array

1	2	3	4	5
4	0	30	5	6

List can be used to store mixed data types. Example

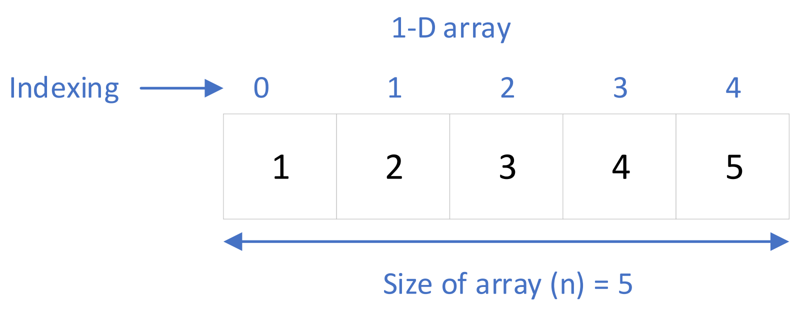
# Mixed data types

x = [1, 1.0, 1+0j, 'one', True]



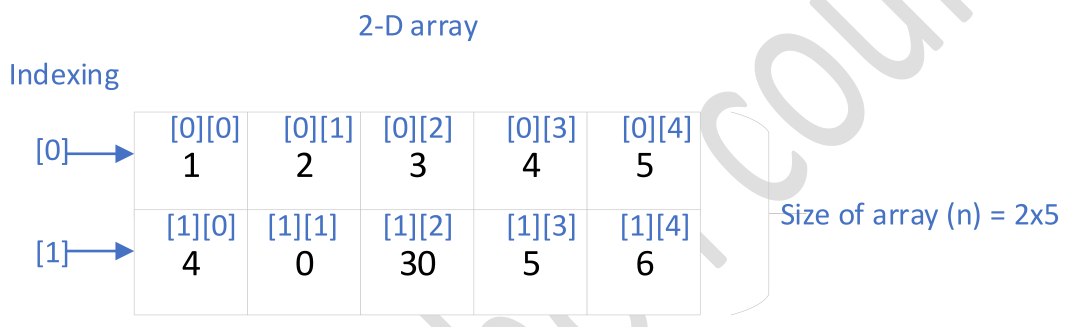
List Manipulation

First you need to understand how indexing an array is done in python. Python uses 0-based indices. For example, size of the 1-D array (n) is 5 which means it has 5 elements xo, x1, z2, x3, x4. Note the indexing tells the position of the elements in the array starting from index 0



For a 2-dimensional array

The size is 2 x5 meaning 2 rows and 5 columns. To index the list in first row you can use list[0] while the second row is list[1]. To index the elements in the first row you use list[0][xi] while to index the elements in the second row you use list[1][xi]. We shall see some examples later on.



Let us create a list of student names. To do that write the codes in your spyder

name = ['john', 'james', 'emma', 'janet', 'musa']

if I am interested in getting the name emma from the list, I can simply write the following code

name_emma = name[2].

print(name_emma)

Note the **name_emma** is a variable name I have just created and the index of 'emma' in the list is 2. So, running that code will store 'emma' in the variable **name_emma**.

Let us take another example. I want to create a list for the following table

Names	**Scores**
john	80
james	75
emma	89
janet	92

musa	68

I can write the following codes

```
Scores = [['john', 'james', 'emma', 'janet', 'musa'],[80, 75, 89, 92, 68]]
```

Now I am interested in knowing the name and score of janet. To do that, I will write the following code

```
name_janet= Scores[0][3]

score_janet = Scores[1][3]

print(name_janet)

print(score_janet)
```

Now try to get the name of james and his score from the list Scores

There are several ways to manipulate a list. They include concatenating, repetition, slicing, and checking if an item is in a list.

Concatenating a List

Simply means to combine elements in a single list

Example

```
L1 = [1,2,3]

L2 = [4,5,6]
```

Now I want to combine the two lists L1 and L2 into one list. I can simply write

```
L3 = L1 + L2

print (L3)

[1, 2, 3, 4, 5, 6]
```

Repetition of a List

Simply means to concatenate multiple copies of the same list

Example

```
L1 = [ 1, 2, 3]

L2 = L1* 4

print(L2)

Out: [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Checking if an item is in the list

We can check if an item is in the list we create. If the item is in the list, it will return a True and if not, it will return a False

```
L1 = [1, 2, 4]

4 in L1

Out: True

10 in L1

Out: False
```

We can check if an item is not in the list. If item is not in the list, it will return True else it will return False

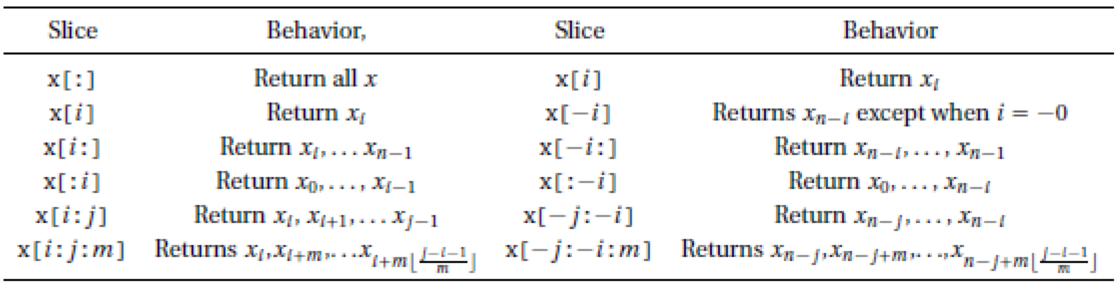
```
L1=[1, 2, 3, 4, 5, 6]

5 not in L1

Out: False
```

Slicing a List

Simply means to return an item of a list at a given index



Slice	Behavior,	Slice	Behavior
x[:]	Return all x	x[i]	Return x_i
x[i]	Return x_i	x[$-i$]	Returns x_{n-i} except when $i = -0$
x[i:]	Return $x_i, \ldots x_{n-1}$	x[$-i$:]	Return $x_{n-i}, \ldots, x_{n-1}$
x[:i]	Return $x_0, \ldots, x_{i-1}$	x[:$-i$]	Return $x_0, \ldots, x_{n-i}$
x[i:j]	Return $x_i, x_{i+1}, \ldots x_{j-1}$	x[$-j$:$-i$]	Return $x_{n-j}, \ldots, x_{n-i}$
x[i:j:m]	Returns $x_i, x_{i+m}, \ldots x_{i+m\lfloor \frac{j-i-1}{m} \rfloor}$	x[$-j$:$-i$:m]	Returns $x_{n-j}, x_{n-j+m}, \ldots, x_{n-j+m\lfloor \frac{j-i-1}{m} \rfloor}$

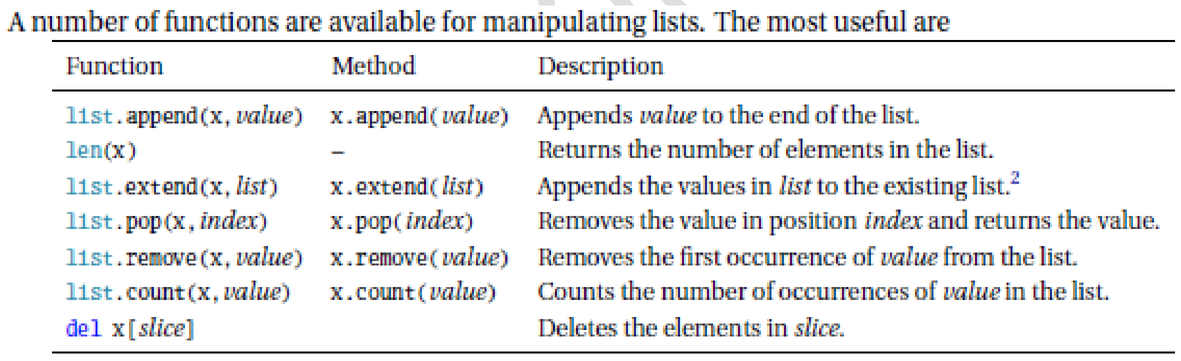
Slice	Example	Remarks
a[i]	a = [3,5,6,7,8,9,7,23,45,67,89] a[0] 3	Returns the element indexed at a_i

	len(a) 11	Note the number of elements (n) in variable **a** is 11
a[i:]	a[3:] [7, 8, 9, 7, 23, 45, 67, 89]	Returns element indexed at $a_3, \ldots \ldots a_{(n-1)}$
a[:]	a[:] [3, 5, 6, 7, 8, 9, 7, 23, 45, 67, 89]	Returns all element in variable **a**
A[1:4]	a[1:4] [5, 6, 7]	Returns elements in variable a_1, a_2, a_3
	Two-dimensional list	

Create a two-dimensional list c	a = [1,3,5,7]
	b= [4 ,5,6,9]
	c = [a,b]
	c
	[[1, 3, 5, 7], [4, 5, 6, 9]]
		Creates a two dimensional list

	c[0] [1, 3, 5, 7]	Returns the first inner list
	>>> c[1] [4, 5, 6, 9]	Returns the second inner list
	>>> c[0][2] 5	Returns the element in the first inner list indexed at 2

List Functions


A number of functions are available for manipulating lists. The most useful are

Function	Method	Description
list.append(x, *value*)	x.append(*value*)	Appends *value* to the end of the list.
len(x)	–	Returns the number of elements in the list.
list.extend(x, *list*)	x.extend(*list*)	Appends the values in *list* to the existing list.[2]
list.pop(x, *index*)	x.pop(*index*)	Removes the value in position *index* and returns the value.
list.remove(x, *value*)	x.remove(*value*)	Removes the first occurrence of *value* from the list.
list.count(x, *value*)	x.count(*value*)	Counts the number of occurrences of *value* in the list.
del x[*slice*]		Deletes the elements in *slice*.

Examples of list manipulations are

Functions	Example	Remarks

append	number = [1,2,3,4,5]
	number.append(9)
	number
	[1, 2, 3, 4, 5, 9]
		# appends 9 to the end of the list

len	len(number) 6	Returns the number of elements in the list - number.

extend	number.extend([20,30,12]) number [1, 2, 3, 4, 5, 9, 20, 30, 12]	Appends the values in list [20.30,12] to the existing list [1, 2, 3, 4, 5, 9]
remove	number.remove(1) number [2, 3, 4, 5, 9, 20, 30, 12]	Removes the first occurrence of value from the list

pop	number = [2, 3, 4, 5, 9, 20, 30, 12]
	number.pop(4)
	
	9
		Removes the value in position index 4 and returns the value 9. Note in python indexing starts from 0.

count	x = [3, 5, 6, 7 , 9 ,10, 7, 2, 7]
	
	x.count(7)
	
	3
		Counts the number of occurrences of value in the list.

del	y = [0,1,2,3,4,5,6,7,8,9] del y[1:3] y [0, 3, 4, 5, 6, 7, 8, 9]	Deletes the elements in slice.

1.5 Application

Now let us write a simple program to record the names and scores of students in a class. To do that, we need to define our variables to store the names and scores of the student. Assuming the names and scores of the students are shown in the Table as follows.

Names	**Scores**
Bill	89
Jane	70
Adrain	72
Segun	100
Hellen	93
Musa	85

```
# first I will create a list for the names and scores
student_names = ['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen', 'Musa']
student_scores = ['89', '70', '72', '100', '93', '85']
# print the student names and scores
print('Table')
print('     Names     Scores')
print('    ',student_names[0], '    ', student_scores[0])
print('    ',student_names[1], '    ', student_scores[1])
print('    ',student_names[2], '  ', student_scores[2])
print('    ',student_names[3], '   ', student_scores[3])
print('    ',student_names[4], '  ', student_scores[4])
```

Result

```
Table
    Names    Scores
    Bill        89
    Jane       70
    Adrain   72
    Segun     100
    Hellen    93
```

Second Let us add a new name and a new score to the existing list Elijah and 90, respectively

```
# I want to append a new name and a new score to the existing list

student_names.append('Elijah')

student_scores.append(90)
```

Result

```
***New addition***

['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen', 'Musa', 'Elijah']

['89', '70', '72', '100', '93', '85', 90]
```

Third Let us add   new list of names and scores to the updated list

```
# I want to extend the current list with 4 other names and scores
new_students = ['Kate', 'Aisha', 'Lisa', 'Angie']
new_scores = [72, 89, 94, 100]
student_names.extend(new_students)
student_scores.extend(new_scores)
# print the latest list
print('***updated  list****')
print(student_names)
print(student_scores)
```

Result

```
***updated list****
['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen', 'Musa', 'Elijah', 'Kate', 'Aisha', 'Lisa', 'Angie']
['89', '70', '72', '100', '93', '85', 90, 72, 89, 94, 100]
```

Now I am interested in finding out the total length of the updated list

```
#find the length of updated list
length_names = len(student_names)
length_scores = len(student_scores)
```

Result

```
Length of student is:  11
Length of scores is:   11
```

Now I am interested in finding out the first 5 names and scores from the latest list

```
# Now I want to extract the first 5 names and the first 5 scores
first_5names = student_names[0:5]
first_5scores = student_scores[0:5]
print(" first five names and scores ")
print(first_5names)
print(first_5scores)
```

Result

```
['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen']
['89', '70', '72', '100', '93']
```

Now I am interested in deleting the last 3 names and last 3 scores from the list

```
del student_names[8:]
del student_scores[8:]
print("**** latest  list ****")
print(student_names)
print(student_scores)
```

Result

```
**** latest  list ****
['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen', 'Musa', 'Elijah', 'Kate']
['89', '70', '72', '100', '93', '85', 90, 72]
```

Note the last three names 'Aisha', 'Lisa', 'Angie' and last three scores 89, 94, 100 have been deleted from the list

Now let is concatenate the list student_name and the list student_score to form a single list

```
# concatenate student_names and student_scores into a single list
student_data = [[student_names],[student_scores]]
print("student data in a single list")
```

```
student data in a single list
[[['Bill', 'Jane', 'Adrain', 'Segun', 'Hellen', 'Musa', 'Elijah', 'Kate']], [['89', '70', '72',
'100', '93', '85', 90, 72]]]
```

1.6 Summary

1. First, we have learnt how to use the python Spyder from the online video
2. We have learnt what a variable is and the rules for creating a variable
3. We have learnt examples of data types and how to manipulate data type list

1.7 Exercise

1. Enter the following into Python, assigning each to a unique variable name:

   (a) 4

   (b) 3.1415

   (c) 1.0

   (d) 2+4j

   (e) 'Hello'

   (f) 'World '

2. Input the variable ex = ’Python is an interesting and useful language for numerical computing!’.

   Using slicing, extract:

   (a) Python

   (b) !

   (c) computing

3. Create a list using the information in the following Table

States	Population
Johor	2000
Sarawak	4000
Sabah	5000
Penang	3000
Perlis	4000

a. Write code to print out the States and population
b. Append the Table as follows to the First Table

States	Population
Kaduna	3000
Jos	2800
Lagos	7000

c. Write a code to delete the following states and their populations
   i. Sarawak and Penang
d. Assuming the value of population of Lagos is incorrect, write a code to replace the value with 6500

Create a file in the spyder editor and save it as exercise1. The file should be uploaded to schoology before 23rd of January.

Please refer to your notes to answer the above questions and also the reference link for more information.

1.8 Reference

[1] http://www.tutorialspoint.com/python/python_basic_operators.htm

[2] https://www.tutorialspoint.com/python/python_basic_operators.htm

Chinese Proverbs.

“Tell me, I forget. Show me, I remember. Involve me, I understand.”