CMP101: Introduction to Computer Science (3 units)

All You Need to Know About Python Programming Language

MR. M. YUSUF

Course Outline

- Introduction
  - Flowcharts
  - How to write Projects Rprt
- Introduction to python
  - Syntax
  - Comments
  - Variables
  - Data types
  - Strings (concat, slice)
  - Numbers
  - Casting
  - Operators/Operands
  - Array []
  - Lists []
  - Tuples ()
  - Dictionaries {}
- Control Structures
  - If… elif…
  - Counting
  - Summing
  - Swapping
  - Max and min
- Loops
  - For Loops
  - While Loops (break, cont. pass)
  - Switch… case
- Functions (arg, param, loc/glob)
  - Built in
  - User define
- OOP: Class, Object, Inheritance
- Modules
- Regular Expressions
- Exceptions
- Files (Read, write, delete)
- Iterators and Collections Modules
- Frameworks
  - GUI,
  - Mobile,
  - Web,
  - IoT,
  - AI

What is Python

- Python is a scripting interpretive Programming Language (PL)
  - Created by Guido van Rossum, and released in 1991
  - Multiplatform, can work on Raspberry P, windows, Mac, Linux etc
  - High Level Language easy to use
  - Rich and available Application Programming Interfaces (API)
  - Procedural, Functional and OOP PL
  - Relies on indentation instead of curly-brackets
  - Versions 2 and 3
- Python is use for
  - Web application development
  - Mobile application development,
  - Artificial Intelligence
  - Internet of Things (IoT).
  - GUI application development

Features of Python

no compiling or linking	rapid development cycle
no type declarations	simpler, shorter, more flexible
automatic memory management	garbage collection
high-level data types and operations	fast development
object-oriented programming	code structuring and reuse, C++
embedding and extending in C	mixed language systems
classes, modules, exceptions	"programming-in-the-large" support
dynamic loading of C modules	simplified extensions, smaller binaries
dynamic reloading of C modules	programs can be modified without stopping

Features of Python Cont.

universal "first-class" object model	fewer restrictions and rules
run-time program construction	handles unforeseen needs, end-user coding
interactive, dynamic nature	incremental development and testing
access to interpreter information	metaprogramming, introspective objects
wide portability	cross-platform programming without ports
compilation to portable byte-code	execution speed, protecting source code
built-in interfaces to external services	system tools, GUIs, persistence, databases, etc.

Features of Python Cont.

		Tcl	Perl	Python	JavaScript	Visual Basic
Speed	development	✓	✓	✓	✓	✓
	regexp	✓	✓	✓		
breadth	extensible	✓		✓		✓
	embeddable	✓		✓		
	easy GUI	✓		✓ (Tk)		✓
	net/web	✓	✓	✓	✓	✓
enterprise	cross-platform	✓	✓	✓	✓	
	I18N	✓		✓	✓	✓
	thread-safe	✓		✓		✓
	database access	✓	✓	✓	✓	✓

Syntax

- ***variable name = input(message to user)***
- num1 = eval(input('Enter the first number: ‘))
- num2 = eval(input('Enter the second number: ‘))
- print('The average of the numbers you entered is', (num1+num2)/2)

Variables

- Rules to follow when naming your variables.
  - Variable names can contain **letters**, **numbers**, and the **underscore**.
  - Variable names **cannot** contain **spaces**.
  - Variable names **cannot start** with a **number**.
  - **Case matters**—for instance, **t**emp and **T**emp are different.

variable name = `input` (*message to user*)

- **temp** = eval(**input**('Enter a temperature in Celsius: '))
- print('In Fahrenheit, that is', 9/5***temp**+32)

String

- *Creating String: s = "Hello"*
- string = input('Enter a string: ') num = eval(input('Enter a number: '))
- Empty string: "" or ''
- Length of string: len('Hello') # 5
- *Concatenation and Repetition*

Expression	Result
'AB'+'cd'	'ABcd'
'A'+'7'+'B'	'A7B'
'Hi'*4	'HiHiHiHi'

- If we want to print a long row of dashes, use print('-'*75)

```
1  print('-'*75)
---------------------------------------------------------------------------
```

String

```
1  s = ''
2  x = int(input("How many letters do you want to check?"))
3  for i in range(x):
4      t = input('Enter a letter: ')
5      if t=='a' or t=='e' or t=='i' or t=='o' or t=='u':
6          s = s + t
7  print(s)
8
```

```
How many letters do you want to check?5
Enter a letter: y
Enter a letter: u
Enter a letter: o
Enter a letter: v
Enter a letter: x
uo
```

String – *in* operator

```
1  name = 'Yusuf Musa'
2  key = 'musa'
3  if key in name:
4    print('Your Name contains '+ key)
5  else:
6    print('Your Name does not contain '+key)
```

```
Your Name does not contain musa
```

String – *indexing*

- *S = "Python"*

Statement	Result	Description
`s[0]`	`P`	first character of `s`
`s[1]`	`y`	second character of `s`
`s[-1]`	`n`	last character of `s`
`s[-2]`	`o`	second-to-last character of `s`

- S[12]
  - ***IndexError: string index out of range***
  - there is only six characters

String – *Slice*

- A slice is used to pick out part of a string.
- Syntax: ***string_name[starting_location : ending_location+1]***
  - Given s='abcdefghij'.

Code	Result	Description
`s[2:5]`	`cde`	characters at indices 2, 3, 4
`s[ :5]`	`abcde`	first five characters
`s[5: ]`	`fghij`	characters from index 5 to the end
`s[-2: ]`	`ij`	last two characters
`s[ : ]`	`abcdefghij`	entire string
`s[1:7:2]`	`bdf`	characters from index 1 to 6, by twos
`s[ : :-1]`	`jihgfedcba`	a negative step reverses the string

String – *Methods*

- Methods is a ***functions*** that ***return*** information about the string or
- return a new string that is a modified version of the original
- **Lower(), upper()**, and **replace()** do not change the original string

Method	Description
`lower()`	returns a string with every letter of the original in lowercase
`upper()`	returns a string with every letter of the original in uppercase
`replace(x,y)`	returns a string with every occurrence of `x` replaced by `y`
`count(x)`	counts the number of occurrences of `x` in the string
`index(x)`	returns the location of the first occurrence of `x`
`isalpha()`	returns `True` if every character of the string is a letter

String – *Methods Cont.*

Statement	Description
`print(s.count(' '))`	prints the number of spaces in the string
`s = s.upper()`	changes the string to all caps
`s = s.replace('Hi','Hello')`	replaces each `'Hi'` in `s` with `'Hello'`
`print(s.index('a'))`	prints location of the first `'a'` in `s`

- **Isalpha()** is used to tell if a character is a **letter** or **not**.

```
1  s = input('Enter a string ')
2  if s[0].isalpha():
3    print('Your string starts with a letter')
4
5  if not s.isalpha():
6    print('Your string contains a non-letter.')
```

```
Enter a string 2yusuf
Your string contains a non-letter.
```

Numbers – Math Operators/order of operation

- 8+5 = 13
- 8-5 = 3
- 8*5 = 40
- 8/5 = 1.6
- 8**5 = 32,768
- 8//5 = 1
- 18%7 = 4

Operator	Description
+	addition
–	subtraction
*	multiplication
/	division
**	exponentiation
//	integer division
%	modulo (remainder)

- PEMDAS
  - Parenthesis
  - Exponentiation
  - Multiplication
  - Division
  - Addition
  - Sally

Numbers – Random Numbers

```
from random import randint
x = randint(1,10)
print('A random number between 1 and 10: ', x)
```

```
A random number between 1 and 10: 7
```

- Using ***randint*** is simple: ***randint(a,b)*** will return a random integer between a and b **including both** a and b.
- (Note that ***randint*** includes the right endpoint b unlike the range function)

Numbers – Math functions

```
from math import sin, pi
print('Pi is roughly', pi)
print('sin(0) =', sin(0))
```

```
Pi is roughly 3.14159265359
sin(0) = 0.0
```

```
print(abs(-4.3))
print(round(3.336, 2))
print(round(345.2, -1))
```

```
4.3
3.37
350.0
```

- Python has a **module** called **math** that contains familiar math **functions**, including **sin**, **cos**, **tan**, **exp**, **log**, **log10**, **factorial**, **sqrt**, **floor**, and **ceil**.
- There are also the inverse trig functions, hyperbolic functions, and the constants **pi** and **e**.
- The round function takes **two arguments**: the first is the **number to be rounded** and the second is the **number of decimal places** to round to.

Numbers – Getting help from Python

- you can type help(math.floor). Typing help(**math**)

```
>>> import math
>>> dir(math)
```

```
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin',
'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos',
'cosh', 'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'hypot', 'isinf', 'isnan', 'ldexp',
'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Numbers – Using the Shell as a Calculator

- The Python shell can be used as a very handy and powerful calculator

```
>>> 23**2
529
>>> s = 0
>>> for n in range(1,10001):
        s = s + 1/n**2

>>> s
1.6448340718480652
>>> from math import *
>>> factorial(10)
3628800
```

Casting

- Python is an OOP, and as such it uses classes to define data types, including its primitive types
  - Casting in python is therefore done using constructor functions int(), float(), and str()

Integer

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

String

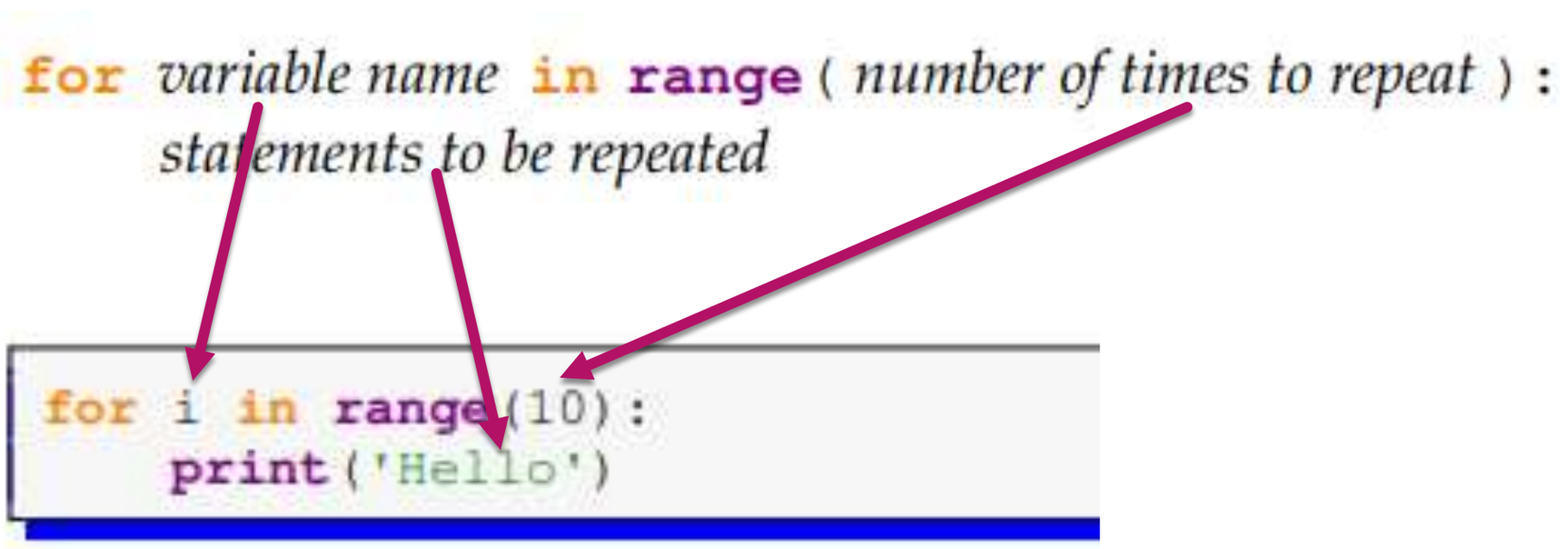
```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Floats

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

For Loops

- While loop allow us to repeat things a **un**specified number of times



- This program will print Hello 10 times

For Loops and range() function

Statement	Values generated
`range(10)`	0,1,2,3,4,5,6,7,8,9
`range(1,10)`	1,2,3,4,5,6,7,8,9
`range(3,7)`	3,4,5,6
`range(2,15,3)`	2,5,8,11,14
`range(9,2,-1)`	9,8,7,6,5,4,3

Ex 1

```
for i in range(3):
    print(i+1, '-- Hello')
```

```
1 -- Hello
2 -- Hello
3 -- Hello
```

Ex 2

```
for i in range(5,0,-1):
    print(i, end=' ')
print('Blast off!!')
```

```
5 4 3 2 1 Blast off!!!
```

For Loops – range() function

Ex 3

```
for i in range(3):
    num = eval(input('Enter a number: '))
    print ('The square of your number is', num*num)
print('The loop is now done.')
```

```
Enter a number: 3
The square of your number is 9
Enter a number: 5
The square of your number is 25
Enter a number: 23
The square of your number is 529
The loop is now done.
```

If… Conditional operators

Expression	Description
`if x>3:`	if x is greater than 3
`if x>=3:`	if x is greater than or equal to 3
`if x==3:`	if x is 3
`if x!=3:`	if x is not 3

If… logical operators

```
if grade>=80 and grade<90:
    print('Your grade is a B.')

if score>1000 or time>20:
    print('Game over.')

if not (score>1000 or time>20):
    print('Game continues.')
```

If … Common Mistakes

S/N	Incorrect	Correct	Remark
1	if x=1:	if x**==**1:	Missing double equality
2	if x>1 and x<100	if x>1 or x<100:	Don't miss exchange
3	if grade>=80 and <90	if grade>=80 and **grade**<90	Missing grade on the RHS
	if grade>=80 and <90	if 80<=grade<90	

…elif… Grade Computation

- Replace the 2nd, 3rd, 4th with elif and the 5th with else
- Only the correct grade will be computed

```
grade = eval(input('Enter your score: '))

if grade>=90:
    print('A')
if grade>=80 and grade<90:
    print('B')
if grade>=70 and grade<80:
    print('C')
if grade>=60 and grade<70:
    print('D')
if grade<60:
    print('F')
```

Counting Ex1

```
count = 0
for i in range(10):
    num = eval(input('Enter a number: '))
    if num>10:
        count=count+1
print('There are', count, 'numbers greater than 10.')
```

Counting Ex2

```
count1 = 0
count2 = 0
for i in range(10):
    num = eval(input('Enter a number: '))
    if num>10:
        count1=count1+1
    if num==0:
        count2=count2+1
print('There are', count1, 'numbers greater than 10.')
print('There are', count2, 'zeroes.')
```

Counting Ex2

```
count = 0
for i in range(1,101):
    if (i**2)%10==4:
        count = count + 1
print(count)
```

While Loops

- We use while loop when **we do not know ahead of time exactly how many times** it has to be repeated or when to stop.

Ex 1

```
for i in range(10):
    print(i)
```

```
i=0
while i<10:
    print(i)
    i=i+1
```

The condition **guess!=secret_num** means that as long as the current guess is not correct, we will keep looping.

Ex 2

```
from random import randint
secret_num = randint(1,10)
guess = 0
while guess != secret_num:
    guess = eval(input('Guess the secret number: '))
print('You finally got it!')
```

Summing

```
s = 0
for i in range(1,101):
    s = s + i
print('The sum is', s)
```

```
s = 0
for i in range(10):
    num = eval(input('Enter a number: '))
    s = s + num
print('The average is', s/10)
```

Swapping

```
x = y
y = x
```

```
hold = x
x = y
y = hold
```

```
x,y = y,x
```

Flag variables

- Used to let one part of your program know when something happens in another part of the program

```
num = eval(input('Enter number: '))

flag = 0
for i in range(2,num):
    if num%i==0:
        flag = 1

if flag==1:
    print('Not prime')
else:
    print('Prime')
```

Maxes and mins

- Used to let one part of your program know when something happens in another part of the program

```
largest = eval(input('Enter a positive number: '))
for i in range(9):
    num = eval(input('Enter a positive number: '))
    if num>largest:
        largest=num
print('Largest number:', largest)
```

Comments

- A comment is a message to someone reading your program.
- Comments are often used to describe what a section of code does or how it works
  - **Single-line comments:** For a single-line comment, use the # character.

```
# a slightly sneaky way to get two values at once
num1, num2 = eval(input('Enter two numbers separated by commas: '))
```

  - **Multi-line comments:** For comments that span several lines, you can use triple quotes.

```
"""
print('This line and the next are inside a comment.')
print('These lines will not get executed.')
"""
print('This line is not in a comment and it will be executed.')
```

Program Comparison

Ex. 1

```
from random import randint

rand_num = randint(5,25)
for i in range(rand_num):
    print('Hello')
```

```
from random import randint

for i in range(randint(5,25)):
    print('Hello')
```

Ex. 2

```
from random import randint

rand_num = randint(1,5)
for i in range(6):
    print('Hello'*rand_num)
```

```
Hello Hello
Hello Hello
Hello Hello
Hello Hello
Hello Hello
Hello Hello
```

```
from random import randint

for i in range(6):
    rand_num = randint(1,5)
    print('Hello'*rand_num)
```

```
Hello Hello Hello
Hello
Hello Hello Hello Hello
Hello Hello Hello
Hello Hello
Hello
```

While loops – infinite loop

- Sometimes a never-ending loop is what you want

```
while True:
    # statements to be repeated go here
```

- In the program below, the value of ***i*** never changes (since ***i*** satisfy the condition ***i<10***) and so the condition ***i<10 always remain true.*** **This program is said to be an infinite loop**

```
i=0
while i<10:
    print(i)
```

For loop with Break Statement

- The break statement can be used to break out of either a ***for*** or ***while loop*** before the loop is finished.

```
for i in range(10):
    num = eval(input('Enter number: '))
    if num<0:
        break
```

```
i=0
num=1
while i<10 and num>0:
    num = eval(input('Enter a number: '))
```

- In each of the program the user is allows to enter up to 10 numbers.
- However, the user can stop early by entering a negative number

While loop and Break Statement

- In each of the program the user can enter the number **-1000** to quite the loop

```
temp = 0
while temp!=-1000:
    temp = eval(input(': '))
    if temp!=-1000:
        print(9/5*temp+32)
    else:
        print('Bye!')
```

```
while True:
    temp = eval(input(': '))
    if temp==-1000:
        print('Bye')
        break
    print(9/5*temp+32)
```

What is the diff. between the two programs?

Ex 1

```
from random import randint
secret_num = randint(1,10)
guess = 0
while guess != secret_num:
    guess = eval(input('Guess the secret number: '))
print('You finally got it!')
```

Ex 2

```
from random import randint

num = randint(1,10)
guess = eval(input('Enter your guess: '))
if guess==num:
    print('You got it!')
```

While loop - The else statement

- There is an optional ***else*** that you can use with ***break*** statements.
- The code indented under the ***else*** gets **executed only if the loop completes** without a break happening.

```
for i in range(10):
    num = eval(input('Enter number: '))
    if num<0:
        print('Stopped early')
        break
else:
    print('User entered all ten values')
```

While and for loop with The else statement

- Here are two ways to check if an integer number is prime.
- A prime number is a number whose only divisors are 1 and itself

```
i=2
while i<num and num%i!=0:
    i=i+1
if i==num:
    print('Prime')
else:
    print('Not prime')
```

```
for i in range(2, num):
    if num%i==0:
        print('Not prime')
        break
else:
    print('Prime')
```

List - []

- Lists can contain all kinds of **datatypes**, even other lists.

```
1  L = eval(input('Enter a list seperated with commer: '))
2  print('The first element is ', L[0])
```

```
Enter a list: 3,4,6,7
The first element is  3
```

- Similarities to strings – **len(), in, index, slice, count, +, *, looping**

Expression	Result
[7,8]+[3,4,5]	[7,8,3,4,5]
[7,8]*3	[7,8,7,8,7,8]
[0]*5	[0,0,0,0,0]

List - [] cont.

Function	Description
`len`	returns the number of items in the list
`sum`	returns the sum of the items in the list
`min`	returns the minimum of the items in the list
`max`	returns the maximum of the items in the list

List – Methods

Method	Description
`append(x)`	adds `x` to the end of the list
`sort()`	sorts the list
`count(x)`	returns the number of times `x` occurs in the list
`index(x)`	returns the location of the first occurrence of `x`
`reverse()`	reverses the list
`remove(x)`	removes first occurrence of `x` from the list
`pop(p)`	removes the item at index `p` and returns its value
`insert(p,x)`	inserts `x` at index `p` of the list

wrong	*right*
`s.replace('X','x')`	`s = s.replace('X','x')`
`L = L.sort()`	`L.sort()`

List – Changing and appending lists

Operation	New L	Description
`L[1]=9`	`[6,9,8]`	replace item at index 1 with 9
`L.insert(1,9)`	`[6,9,7,8]`	insert a 9 at index 1 without replacing
`del L[1]`	`[6,8]`	delete second item
`del L[:2]`	`[8]`	delete first two items

```
1  from random import randint
2  L = []
3  for i in range(50):
4    #L.append(randint(1,100))
5    L = L + [randint(1,100)]
6
7  print(L)
8
9
```

```
[40, 22, 86, 52, 76, 83, 96, 60, 11, 34, 74, 1, 46, 88, 42, 66, 86, 77, 18, 16, 67, 10, 57, 34, 65, 26, 19, 74, 51, 31, 40
```

List – Code Examples

```
1   scores = [40, 22, 86, 52, 76, 83, 96, 60, 11, 34, 74, 1, 46, 88, 42, 66]
2   scores.sort()
3   print(scores)
4   print('Two smallest: ', scores[0], scores[1])
5   print('Two largest: ', scores[-1], scores[-2])
```

```
[1, 11, 22, 34, 40, 42, 46, 52, 60, 66, 74, 76, 83, 86, 88, 96]
Two smallest:  1 11
Two largest:  96 88
```

List – interactive Game

```
num_right = 0
# Question 1
print('What is the capital of Nigeria?', end=' ')
guess = input()
if guess.lower()=='abuja':
  print('Correct!')
  num_right+=1
else:
  print('Wrong. The answer is Paris.')
  print('You have', num_right, 'out of 1 right')

#Question 2
print('Which state are neigbor to abuja? ', end=' ')
guess = input()
if guess.lower()=='kaduna' or guess.lower()=='niger' or guess.lower()=='nasarawa':
  print('Correct!')
  num_right+=1
  print('You have ', num_right, 'out of 2 right,')
else:
  print('Wrong. The answer is Kaduna, Niger, or Nasarawa.')
  print('You have ', num_right, 'out of 2 right ,')
```

```
What is the capital of Nigeria? yola
Wrong. The answer is Paris.
You have 0 out of 1 right
Which state are neigbor to abuja?  kano
Wrong. The answer is Kaduna, Niger, or Nasarawa.
You have  0 out of 2 right ,
```

```
num_right = 0
# Question 1
print('What is the capital of Nigeria?', end=' ')
guess = input()
if guess.lower()=='abuja':
  print('Correct!')
  num_right+=1
else:
  print('Wrong. The answer is Paris.')
  print('You have', num_right, 'out of 1 right')

#Question 2
print('Which state are neigbor to abuja? ', end=' ')
guess = input()
if guess.lower()=='kaduna' or guess.lower()=='niger' or guess.lower()=='nasarawa':
  print('Correct!')
  num_right+=1
  print('You have ', num_right, 'out of 2 right,')
else:
  print('Wrong. The answer is Kaduna, Niger, or Nasarawa.')
  print('You have ', num_right, 'out of 2 right ,')
```

```
What is the capital of Nigeria? abuja
Correct!
Which state are neigbor to abuja?  kaduna
Correct!
You have  2 out of 2 right,
```

List – interactive Game

```
1   questions = ['What is the capital of Nigeria? ','Whats the name of US president? ']
2   answers = ['Abuja','Biden']
3   num_right = 0
4   for i in range(len(questions)):
5     guess = input(questions[i])
6     if guess.lower()==answers[i].lower():
7       print('Correct')
8       num_right=num_right+1
9       print('You have', num_right, 'out of', i+1, 'right.')
10    else:
11      print('Wrong. The answer is', answers[i])
12      print('You have', num_right, 'out of', i+1, 'right.')
13
```

```
What is the capital of Nigeria? lagos
Wrong. The answer is Abuja
You have 0 out of 1 right.
Whats the name of US president? biden
Correct
You have 1 out of 2 right.
```

Booleans

- **Boolean** variables in Python are variables that can take on two values, **True** and **False**
  - Help make your programs more readable.
  - Used as flag variables or to indicate options

```
if game_over:          ⇔   if game_over==True:
while not game_over:   ⇔   while game_over==False:
```

  - **Conditional expressions** evaluate to **Booleans** and you can even assign them to **variables**. E.g. **x = (6==6)** will evaluate to **True**

Python Shortcuts

1. Operator Shortcuts

Statement	Shorthand
`count=count+1`	`count+=1`
`total=total-5`	`total-=5`
`prod=prod*2`	`prod*=2`

2. Shortcuts with conditions

Statement	Shortcut
`if a==0 and b==0 and c==0:`	`if a==b==c==0:`
`if 1<a and a<b and b<5:`	`if 1<a<b<5:`

3. Variable declaration and initialization Shortcut

```
x = L[0]
y = L[1]
z = L[2]
```

=

```
x,y,z = L
```

or

```
a = 0
b = 0
c = 0
```

=

```
a = b = c = 0
```

4. Swapping Shortcut

```
x,y,z = 1,2,3
```

=

```
x,y,z = y,z,x
```

=

Thank You