



# Covenant University

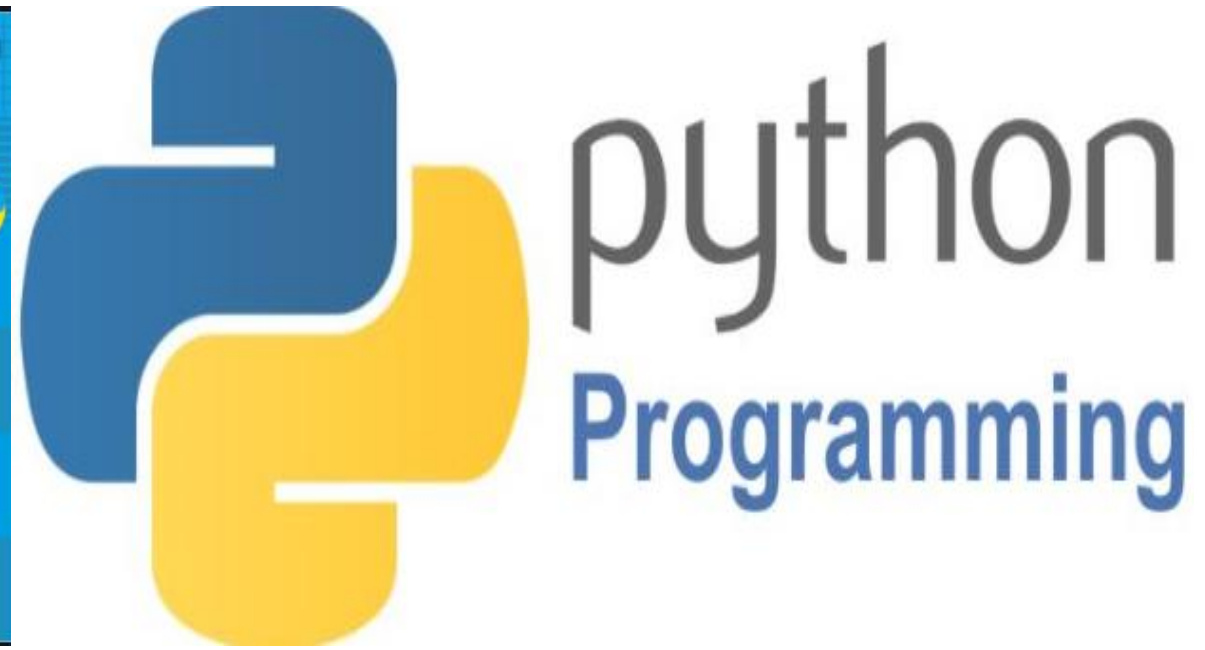
Raising a new Generation of Leaders

## PET328

### COMPUTER APPLICATIONS IN PETROLEUM ENGINEERING

# PET328: COMPUTER APPLICATIONS IN PETROLEUM ENGINEERING

## (With Python Programming)







*Olatunde O. Mosobalaje (PhD)*





Department of Petroleum Engineering,  
Covenant University, Ota  
Nigeria

# OUTLINE

## Preambles

-  The Appetizer
-  The Toolbox
-  The Embedded Course
-  Introduction to Computer Programming

## Getting Started with Python

-  Basic Python Objects
-  Conditional Execution
-  Repeated Execution
-  Functions

## Python Data Structures

-  Strings
-  Lists
-  Tuples
-  Dictionaries

## Application Projects

-  Oil Reservoir Volumetrics
-  Material Balance Analysis
-  PVT Properties





# PREAMBLES

## The Embedded Course

A Coursera course is embedded into this course (PET328). It is compulsory that all students completes the Coursera course as it is part of the assessment items in PET328.



## PREAMBLES

### The Embedded Course

---

The embedded course is titled ‘Programming for Everybody (Getting Started with Python)’.

The course is offered by University of Michigan.

Programming for Everybody (Getting Started with Python)

Offered by



## PREAMBLES

### The Embedded Course

The link to the embedded course has been added to the PET328 course site on Moodle.

To enroll for the Coursera course, simply click on the link.

✚ 28 March - 3 April ✎

✚  Coursera Course - Programming for Everybody ✎

Getting Started with Python

# PREAMBLES

## The Embedded Course



### Programming for Everybody (Getting Started with Python)

★★★★★ 4.8 189,529 ratings • 45,380 reviews

[Go to Course](#)



Save for Later

Sponsored by Covenant University

#### About this Course

This course aims to teach everyone the basics of programming computers using Python. We cover the basics of how one constructs a program from a series of

Offered by



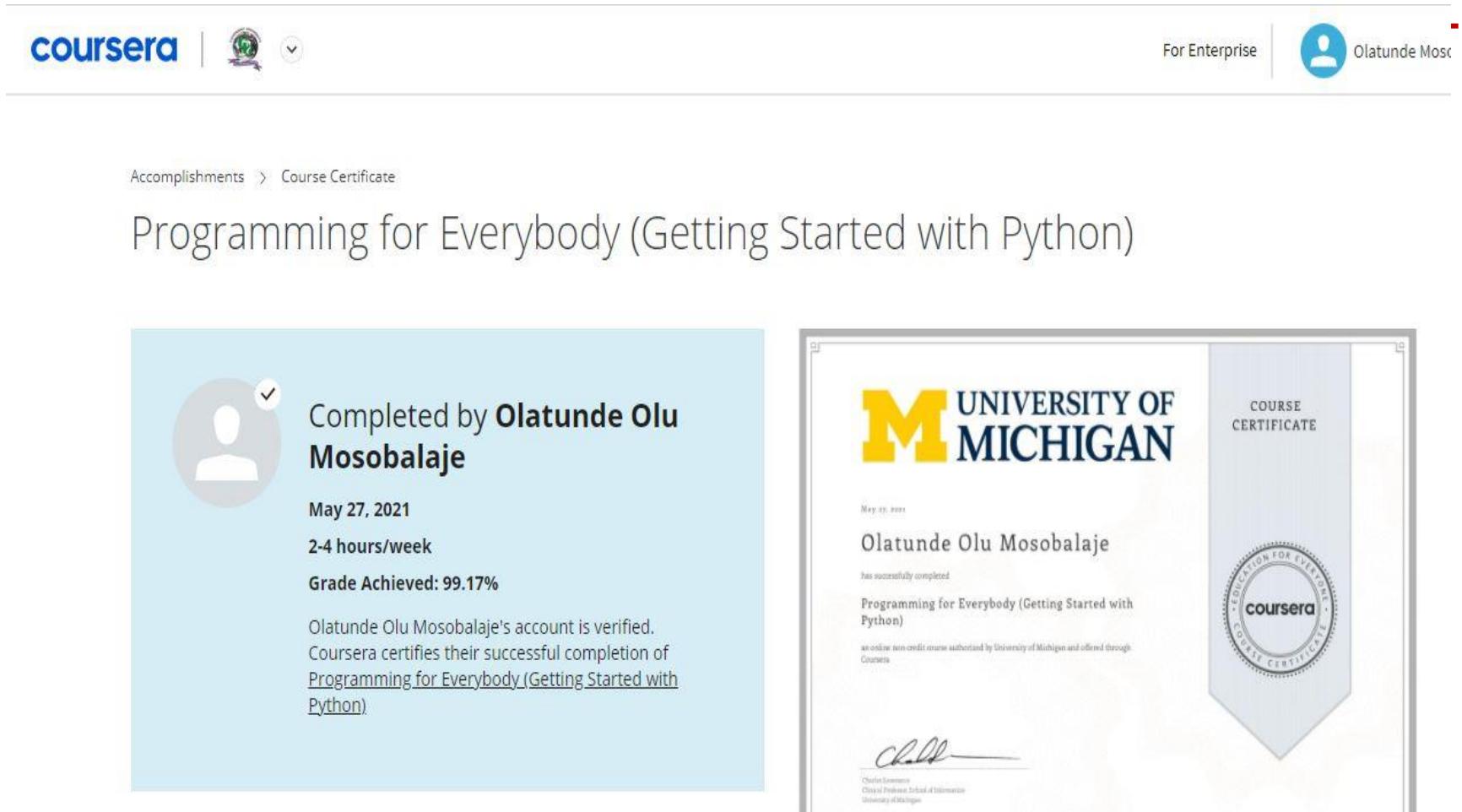
#### Flexible deadlines

Reset deadlines in accordance to your schedule.

# PREAMBLES

## The Embedded Course

When you complete the course, you are awarded a certificate of completion!!!



The screenshot shows the Coursera interface for a completed course. At the top, the Coursera logo and a user profile icon for 'Olatunde Mosobalaje' are visible. The main heading is 'Programming for Everybody (Getting Started with Python)'. Below this, a light blue box displays the completion details: 'Completed by Olatunde Olu Mosobalaje', 'May 27, 2021', '2-4 hours/week', and 'Grade Achieved: 99.17%'. A note states: 'Olatunde Olu Mosobalaje's account is verified. Coursera certifies their successful completion of Programming for Everybody (Getting Started with Python).' To the right, a 'COURSE CERTIFICATE' from the 'UNIVERSITY OF MICHIGAN' is shown, dated 'May 27, 2021', awarded to 'Olatunde Olu Mosobalaje' for completing the course. The certificate includes the Coursera logo and a signature line.



## PREAMBLES

### Introduction to Computer Programming

- 🔗 Analogy: Programming language vs. Natural language
  - 🔗 There is a striking similarity between learning programming language and learning natural language. In both cases, the process is thus:
    - 🔗 learn the vocabulary and the grammar – spell words, construct sentences etc.
    - 🔗 Communicate
      - 🔗 natural language: use words, sentences, paragraphs to communicate an idea
      - 🔗 Programming language: use **keywords**, **variables**, **functions**, **expressions**, **statements** to communicate steps to computer.

# PREAMBLES

## Introduction to Computer

## Programming

🐍 A program is simply a collection of sequential Python statements written to perform a specific task

```
def bubble_sort(list):
    sorted_list = list[:]
    is_sorted = False
    while is_sorted == False:
        swaps = 0
        for i in range(len(list) - 1):
            if sorted_list[i] > sorted_list[i + 1]: # swap
                temp = sorted_list[i]
                sorted_list[i] = sorted_list[i + 1]
                sorted_list[i + 1] = temp
                swaps += 1
        print(swaps)
        if swaps == 0:
            is_sorted = True
    return sorted_list

print(bubble_sort([2, 1, 3]))
```

## PREAMBLES

### Introduction to Computer Programming

Fundamental patterns (concepts) in a program

✚ The following are typical patterns (statement(s)) you see in a program:

- ✚ Input statements
- ✚ Output statements
- ✚ Sequential execution
- ✚ Conditional statements
- ✚ Repeated execution (loops)
- ✚ Reuse of statements (functions)

## PREAMBLES

## Introduction to Computer Programming

Fundamental patterns (concepts) in a program

- Input statements: - used to request and accept data from users
- Example: the `input` function.

```
Input_Output_demo.py - C:\Users\TTOWG\645\1 karia def\2. CU\CU Courses\PET328 - Computer Applications in Petroleum
File Edit Format Run Options Window Help
#...TTOWG!

# input statements
poro = input('Enter the value of porosity: ')
area = input('Enter the value of area: ')
paythickness = input('Enter the value of pay zone thickness: ')

area = float(area)
paythickness = float(paythickness)

BV = area*paythickness

# output statement
print('The bulk volume of the reservoir is', BV, 'Acre-ft')
```

This script is available [here](https://github.com/TTOWG/PET328_2021_Class/blob/main/demo_I_input_Output.py)

[https://github.com/TTOWG/PET328\\_2021\\_Class/blob/main/demo\\_I](https://github.com/TTOWG/PET328_2021_Class/blob/main/demo_I_input_Output.py)

[input\\_Output.py](https://github.com/TTOWG/PET328_2021_Class/blob/main/demo_I_input_Output.py)



# PREAMBLES

## Introduction to Computer Programming

Fundamental patterns (concepts) in a program

🔗 Output statements: - used to display the results of execution on the screen.

🔗 Example: the **print** function

```
Input_Output_demo.py - C:\Users\TTOWG\645\1 karia def\2. CU\CU Courses\PET328 - Computer Applications in Petroleum
File Edit Format Run Options Window Help
#...TTOWG!

# input statements
poro = input('Enter the value of porosity: ')
area = input('Enter the value of area: ')
paythickness = input('Enter the value of pay zone thickness: ')

area = float(area)
paythickness = float(paythickness)

BV = area*paythickness

# output statement
print('The bulk volume of the reservoir is', BV, 'Acre-ft')
```

# PREAMBLES

## Introduction to Computer

## Programming

Fundamental patterns (concepts) in a program

- Sequential execution: - typically, a program would entail multiple statements.
- Statements are executed in the order (sequence) in which they are encountered.
- Latter statements can make use of results of former statements; not vice-versa

```

Input_Output_demo.py - C:\Users\TTOWG\645\1 karia def\2. CU\CU Courses\PET328 - Computer Applications in Petroleum
File Edit Format Run Options Window Help
#...TTOWG!

# input statements
poro = input('Enter the value of porosity: ')
area = input('Enter the value of area: ')
paythickness = input('Enter the value of pay zone thickness: ')

area = float(area)
paythickness = float(paythickness)




BV = area*paythickness

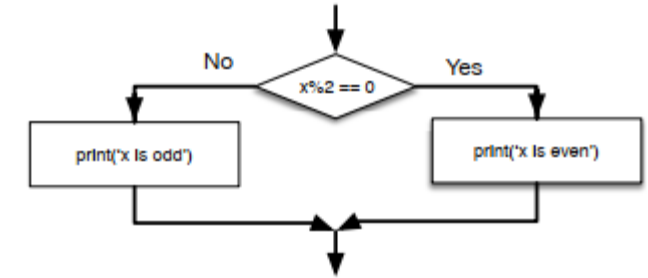
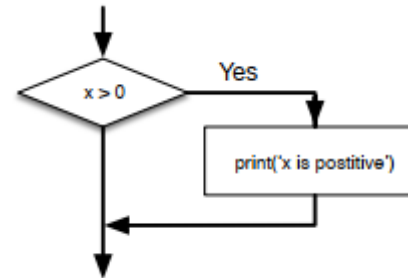
# output statement
print('The bulk volume of the reservoir is', BV, 'Acre-ft')
    
```

# PREAMBLES

## Introduction to Computer Programming

Fundamental patterns (concepts) in a program

-  Conditional Statements: patterns that make it possible for the program to **check** for some conditions and **decide** to:
  -  perform a statement(s) or skip the statement(s)
  -  Choose between alternative statements.



```

conditional_statement_demo.py - C:\Users\TTOWG\645\1 karia def\2. CU\CU Courses\PET328 - Computer Applications in Petroleum Enginee
File Edit Format Run Options Window Help
#...TTOWG!

initial_pressure = input('Enter the value of initial pressure: ')
bubble_pressure = input('Enter the value of bubble-point pressure: ')

initial_pressure = float(initial_pressure)
bubble_pressure = float(bubble_pressure)

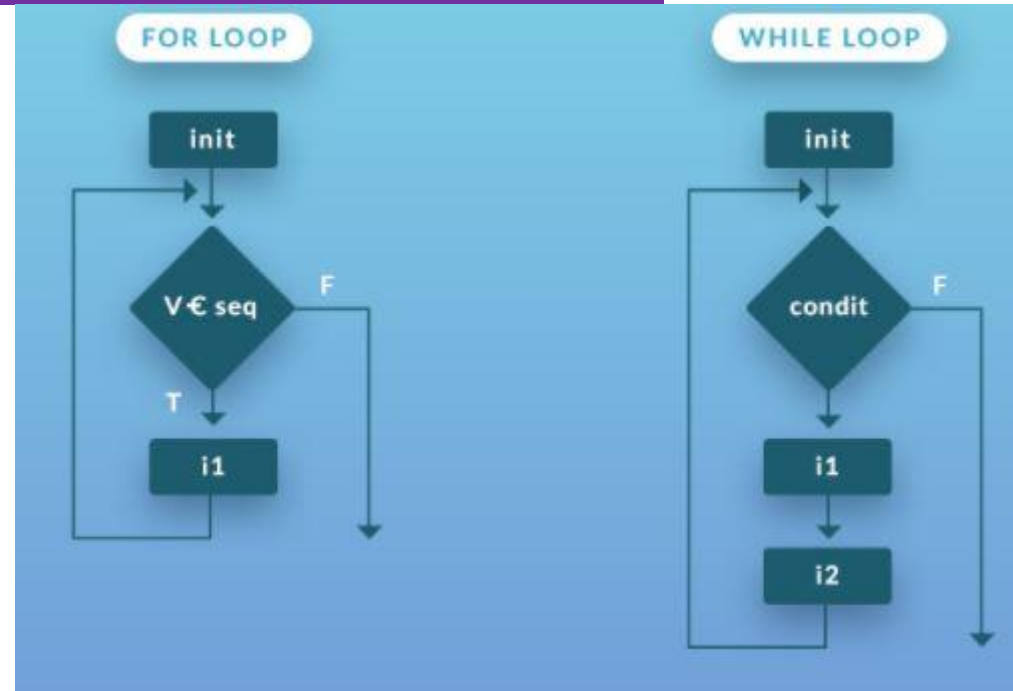
if initial_pressure > bubble_pressure:
    print('The reservoir is undersaturated!!!')
else:
    print('The reservoir is saturated!!!')
    
```

# PREAMBLES

## Introduction to Computer Programming

Fundamental patterns (concepts) in a program

- Repeated Execution: patterns that instructs the program to perform (iterate) a given statement(s) repeatedly, for each item in a set of items, varying values of parameter(s) from item to item.



```
repeated_execution_demo.py - C:/Users/TTOWG/645/1 karia def,
File Edit Format Run Options Window Help
#...TTOWG!

blocks = [1,2,3,4,5]
for block in blocks:
    print('This is Block', block)
```



## PREAMBLES

## Introduction to Computer

## Programming

## Fundamental patterns (concepts) in a program

- Re-use of statements: the task performed by some statement(s) might be routine and needed at various points in your program.
- Such statement(s) may be written once, saved with a name and re-used at various points in your program by referring to the name.

```
statement_reuse_demo.py - C:\Users\TTOWG\645\1 karia def\2. CU\CU Courses\PET328 - Computer Applications in Petroleum Engineering\Demos\statement_reuse_de...
File Edit Format Run Options Window Help
#...TTOWG!

# function definition
def stoiip_calc(area, thickness, poro, sw, boi):
    STOIP = (7758*area*thickness*poro*(1-sw))/boi
    return STOIP

# function call for Reservoir TTOWG_1 (re-use)
oil_inplace_TTOWG_1 = stoiip_calc(40, 15, 0.3, 0.28, 1.2)
print('The amount of oil in place in Reservoir TTOWG_1 is', oil_inplace_TTOWG_1, 'STB')

# function call for Reservoir TTOWG_2 (re-use)
oil_inplace_TTOWG_2 = stoiip_calc(80, 10, 0.23, 0.35, 1.1)
print('The amount of oil in place in Reservoir TTOWG_2 is', oil_inplace_TTOWG_2, 'STB')
```

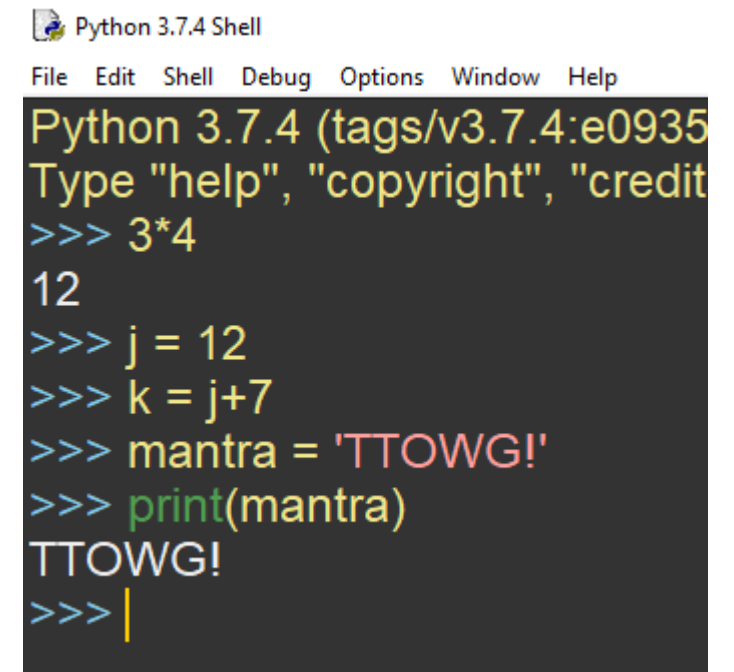
```
>>>#TTOWG!
```

```
>>>print('...to the only wise God')
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

- Crudely speaking, Python objects are stuffs upon which actions (specified in python commands) are performed.
- Example: in the code screenshot shown, 3, 4, j, k, 7, mantra, 'TTOWG' are all objects acted upon.



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e0935f6, Dec 16 2019)
Type "help", "copyright", "credits()" or "quit()" for more
>>> 3*4
12
>>> j = 12
>>> k = j+7
>>> mantra = 'TTOWG!'
>>> print(mantra)
TTOWG!
>>> |
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

- ✚ The basic Python objects considered here are  
Values and Variables.
- ✚ Later, some sets of sophisticated objects  
known as data structure shall be considered.



# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Values

- Values are simply the representation of data entities.

### Types of Values

- Values in Python belong to various types such as type *integer*, type *float*, and type *string*.
- Use the function *type* to find out the type to which a value belong.

```
>>> type(2)
<class 'int'>
>>> type('TTOWG!')
<class 'str'>
>>> type(2.0)
<class 'float'>
>>> type('2')
<class 'str'>
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Types of Values

- 2 is of type (class) integer
- 'TTOWG' and '2' are of type string; just like any set of characters (alphanumeric and non-alphanumeric) enclosed in quotes
- 2.0 is of type float; just as are all numbers expressed in decimals.

```
>>> type(2)
<class 'int'>
>>> type('TTOWG!')
<class 'str'>
>>> type(2.0)
<class 'float'>
>>> type('2')
<class 'str'>
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Types of Values

- ☛ Please, take note that users' response to the input function prompt is stored as a string.
- ☛ Before using such *input* in a mathematical operation, they should be converted to a numerical type using *float* or *int* functions.

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22)
Type "help", "copyright", "credits" or "license()" for more infor
>>> poro = input('What is the value of porosity?')
What is the value of porosity?0.34
>>> print(poro)
0.34
>>> type(poro)
<class 'str'>
>>> poro/0.01
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    poro/0.01
TypeError: unsupported operand type(s) for /: 'str' and 'float'
>>> # The division operation failed because
>>> # the value 0.34 is a string; not a number.
>>> poro = float(poro)
>>> type(poro)
<class 'float'>
>>> poro/0.01
34.0
>>> |
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Variables

- 🔗 A variable is a value stored in memory and referred to with a chosen name.
- 🔗 In other words, values are assigned to variables.
- 🔗 When the name of a variable is called, the value assigned therein answers.

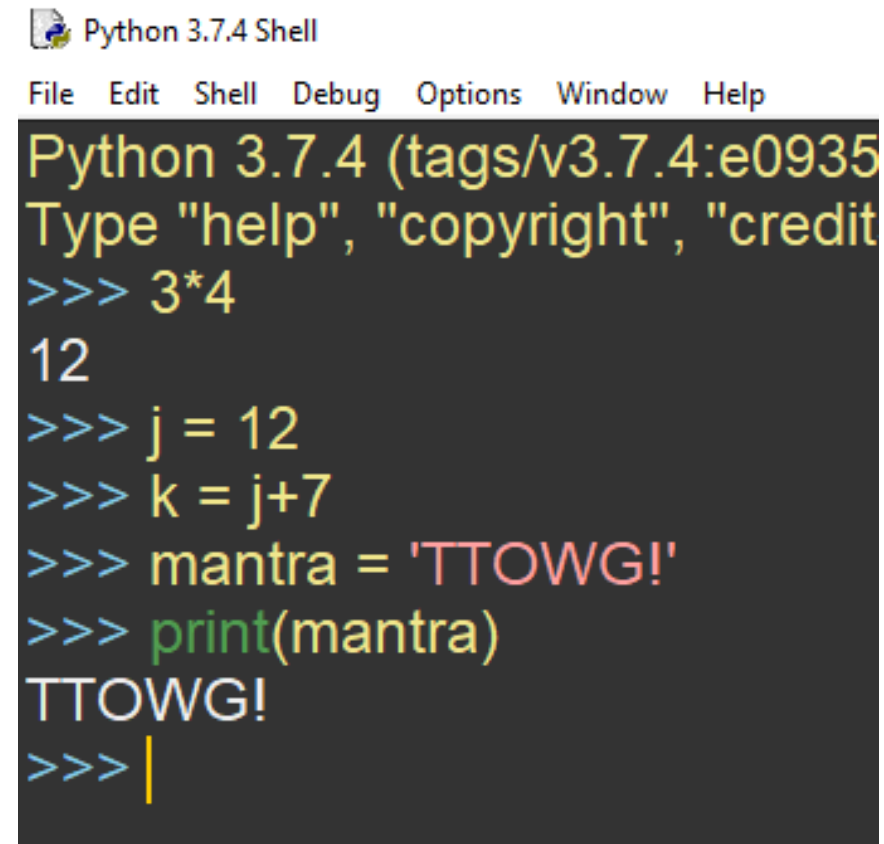


# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Variables

- ✚ A raw value can be assigned to a variable.
  - ✚ Example: j is a variable; the value 12 is assigned to it.
- ✚ Also, the output of an expression (involving a variable) may be stored in another variable.
  - ✚ Example: k is a variable, the value obtained when  $j+7$  is executed is subsequently assigned to variable k.
- ✚ Not only numeric values are assigned to variables, strings are also assigned.
  - ✚ Example, mantra is a variable with string "TTOWG!" assigned to it.



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e0935
Type "help", "copyright", "credit
>>> 3*4
12
>>> j = 12
>>> k = j+7
>>> mantra = 'TTOWG!'
>>> print(mantra)
TTOWG!
>>> |
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Choosing Variable Names

- ✚ In naming variable, the following rules are recommended:
  - ✚ Variable names should be descriptive, as much as possible. That is, the name should somewhat tell us something about the variable. Example: a variable to hold the value of reservoir permeability is better named 'perm' than named 'x'
  - ✚ The name must be a single word. Where multiple words are necessary for descriptive purposes, they can be joined with the underscore character; e. g.: init\_pressure.
  - ✚ Names should not be too long.
  - ✚ Names may contain both alphabets and numbers; but must not start with numbers.
  - ✚ Names are case sensitive. If you named a variable as 'poro', do not refer to it as 'Poro'.
  - ✚ Avoid using special characters like '@', '\$' in names.

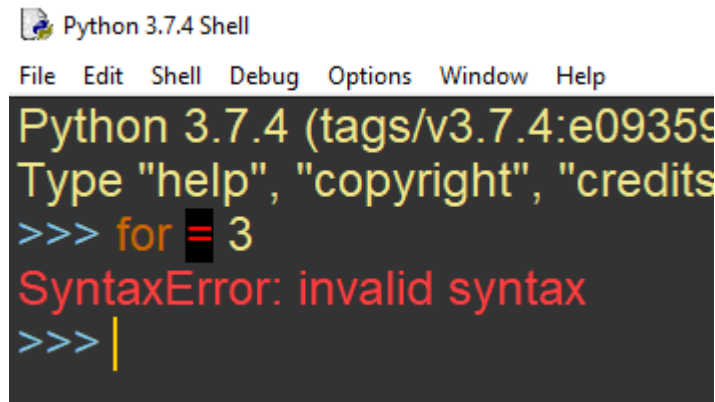
# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Keywords

- Keywords are words that are reserved for Python's in-built structure.
- Here is the list of Python's keywords.
- Keywords cannot be used as variable names; doing so would cause error.

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



```

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e093596, Oct 14 2019)
Type "help", "copyright", "credits" or "quit()"
>>> for = 3
SyntaxError: invalid syntax
>>> |
    
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Statements

- A statement is simply unit of code (commands) that is interpretable and executable by Python; just like a sentence in natural language.
- Two common types of Python statements are **Assignment statements** and **Expressions**.
- Assignment statements simply assigns values to a variable.
- An expression is a statement that combines variables, values, functions and operators.
- A statement could combine both types such that the result of an expression (RHS) is assigned to a variable (LHS).

```
poro = 0.27 # This is an assignment statement.
```

```
area = 40 # This is an assignment statement.
```

```
thickness = 15 # This is an assignment statement.
```

```
area*thickness # This is an expression.
```

```
PV = area*thickness*poro # A combination of expression (RHS) and assignment
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Multi-line statements

- Typically, a Python statement is written in a single line.
- However, if the statement is too long, it could be continued in the next line; but the current line should end with the line continuation character i.e. \

```
>>> 17+2+9 \
      +3+23
54
>>> |
```

### Multiple statements in a line

- Writing multiple statements in same line is not encouraged; however, if that has to be done, the statements should be separated by semicolon.

```
>>> poro = 0.18; area = 40; thickness = 15
>>> print(area)
40
>>> |
```



# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Operators

🔗 Operators are symbols of mathematical operations.

🔗 + for addition

🔗 - for subtraction

🔗 \* for multiplication

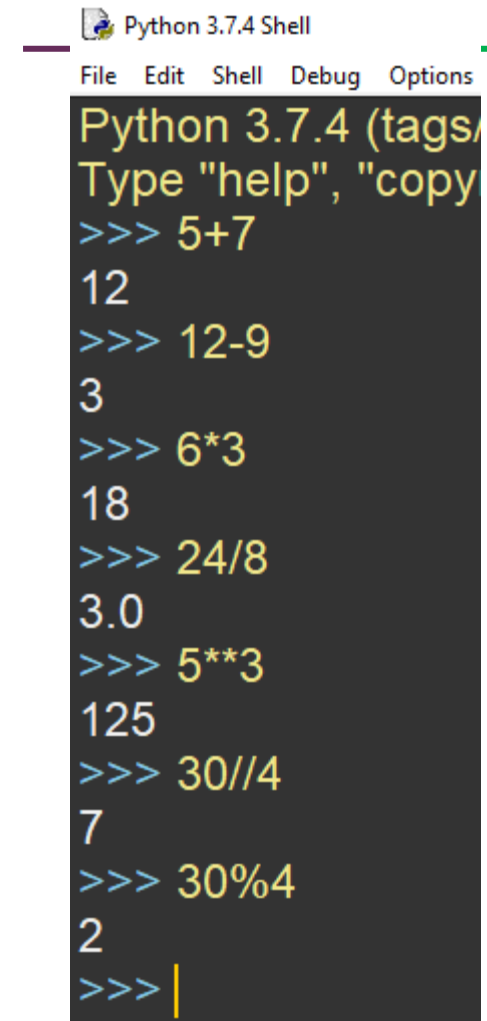
🔗 / for division

🔗 \*\* for exponentiation (raise to power)

🔗 // integer division (truncates the result of division to its integer part.

🔗 % modulus (gives the remainder of an integer division).

🔗 The objects acted upon by operators are called operands.



```
Python 3.7.4 Shell
File Edit Shell Debug Options
Python 3.7.4 (tags/
Type "help", "copy
>>> 5+7
12
>>> 12-9
3
>>> 6*3
18
>>> 24/8
3.0
>>> 5**3
125
>>> 30//4
7
>>> 30%4
2
>>> |
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Order of Operations

- When multiple operations are featured in a statement, Python executes them in the order specified by the acronym: PE-MD-AS (Parenthesis, Exponentiation, Multiplication, Division, and Subtraction).
- You can use parenthesis to dictate the order you desire.
- Multiplication and Division has equal precedence; hence are executed left to right.
- Addition and Subtraction has equal precedence; hence are executed left to right.
- You may also use parenthesis to make an expression more readable and less confusing.
- Nested parenthesis are executed from inside to outside.

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### Order of Operations

Consider the following operations to convince yourself of the PEMDAS order.

$( )$	$\sqrt{x} \ x^2$	$\times$ OR $\div$	$+$ OR $-$		
Parentheses	Exponents	Multiply	Divide	Add	Subtract
P	E	M	D	A	S

$$\begin{aligned}
 & 3 \times (3 + 7) - 4^2 \div 2 \\
 &= 3 \times 10 - 4^2 \div 2 \\
 &= 3 \times 10 - 16 \div 2 \\
 &= 30 - 8 \\
 &= 22
 \end{aligned}$$

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1905 64-bit (AMD64)]  
Type "help", "copyright", "credits" or "license()" for more information.

```
>>> (3+5)**(9-6)
```

```
512
```

```
>>> 14+(3+5)**(9-6)
```

```
526
```

```
>>> # No, I mean the result of 14+(3+5) should be raised to power 9-6
```

```
>>> # Oh! Use parenthesis to dictate that order:
```

```
>>> (14+(3+5))**(9-6)
```

```
10648
```

```
>>> 14+(3+5)**(9-6)/10
```

```
65.2
```

```
>>> (14+(3+5))**(9-6)/10
```

```
52.6
```

```
>>> (14+(3+5))**(9-6)/10
```

```
1064.8
```

```
>>>
```

# GETTING STARTED WITH PYTHON

## Basic Python Objects

### String Operations

- Strings can be joined end-to-end by using the + operator. If you want a space between the strings, then include it in one of the strings.
- Also, a string can be repeated multiple times using the \* operator (with an integer, of course).

```
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19
Type "help", "copyright", "credits" or "license()" for mo
>>> 'TTOWG!' + 'to the only wise God'
'TTOWG!to the only wise God'
>>> # Oh, I need space
>>> 'TTOWG!' + 'to the only wise God'
'TTOWG! to the only wise God'
>>>
>>> 'TTOWG!'*3
'TTOWG!TTOWG!TTOWG!'
>>> 'TTOWG!'*3
'TTOWG! TTOWG! TTOWG! '
>>> 3*'TTOWG!'
'TTOWG! TTOWG! TTOWG! '
>>> |
```

```
>>>#TTOWG!
```

```
>>>print('...to the only wise God')
```