Scientific Programming with Python

Jerry Ebalunode

UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

Today's Lecture: Introduction to Python

- Python Interactive Shell
- Writing and running scripts
- Objects
- Assignments
- Variables
- How to request input from users
- Processing input
- How to send output to terminal/screen

JupyterHub
An online learning platform for Python

- We will be using jupyterhub for the class today
- The tool provides access to:
  - Python, command line etc
- To connect you need → UH cougarnet iD and password
- Link → https://neches.rcdc.uh.edu
  - You can connect directly from anywhere on UH network
  - From Outside network use UH VPN

Alternate Access to Python Notebook via Anaconda

- Several prepackaged Python distributions available
- One of the most popular ones is Anaconda Python
  - https://www.anaconda.com
  - Installed currently in HPEDSI teaching and cluster infrastructure
- Students can install anaconda python on there laptop
  - https://www.anaconda.com/distribution/#download-section
  - Pick the versions compatible with your computer.

Alternate Access to Python notebook via Collab

- Google Collab
  - Google account needed
- https://colab.research.google.com
  -
- Provides access to python notebooks and much more online
  - No installation required
  - Free accounts for limited hardware use
  - Short introduction available video
  - https://youtu.be/inN8seMm7UI

Basics of Python

- Python is a dynamic, multi-purpose, object-oriented, scripted programming language
  - High-level language
    - built-in high-level data types i.e. flexible arrays, dictionaries etc
  - Extensible
    - new code, extends objects & definition, or system type modification at runtime.
- Developed by Guido van Rossum
  - Early 1990s, Netherlands
- Named after the BBC show “Monty Python’s Flying Circus”
  - nothing to do with reptiles
- Currently two main branches
  - ~~Python 2.x now at version 2.7.18~~

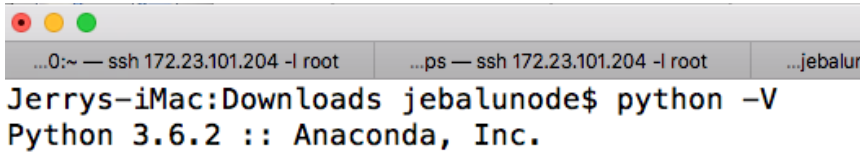
    (End of Life after 1/1/2020, not maintained afterwards)
  - Python 3.x now at version 3.7.13, 3.8.13, 3.9.13, 3.10.6
    - Python 3.7 enjoy larger code base support
    - Library support
      - 3.7 > (better than) 3.8 > (better than) 3.8 > 3.9 > 3.10
  - Note Python <=3.6 last update in Sept 2021
- Official URL http://python.org
- Open source software (OSS)

The Python Interactive Interpreter

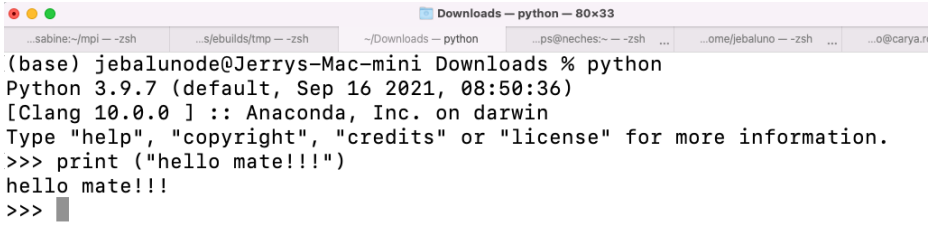
1. Open a terminal in your computer
   - Mac users:
     - →Finder →Applications →Utilities →Terminal
   - Windows users:
     - →Start →Program Files →Accessories →Command Prompt
2. Type can type python
3. How to figure out which version of python you have?
   - Type *python -V* in the terminal window

The Python Interactive Interpreter

Checking python version



python in interactive mode



Python Interactive Shell

- Commands tell the python interpreter to do something. For example type the following sequence of statements in the python shell:

```
print("This is an example of a print statement")
print('This is another example of a print statement')
print("""The print built in function prints the set of
    values given as input""")
print(5*8)
```

Writing Programs in Python

- Python programs have the file extension ".py"
- Computer programs are "case sensitive"
  - **Print** is not equal to **print**
- Remember to add quotations assigning string literals or commenting
  - **In python,** " "
  - **are equivalent to ' ', and to """ """ but they have to match**

Executing Programs In Python

- Two ways to execute your python programs:

1. Create a text file containing your python code
   i.e. my_program.py

2. Open a terminal window, go to the directory where 'my_program.py' is located and type:
   $ python my_program.py

Objects

- Core elements that Python manipulates
- Also called value
- Each object has a type, or class, that defines the set of operations, and data available for objects of that type.
- Python has the built-in function type that can tell you the class of that object:

```
print(type("What type is this?"))
print(type(5*8))
```

- So for now, just know that Python is always dealing with **objects**

Basic Data Types in Python

Type	Python Type	Examples
Integers	int	123, -123, 0
Real valued (decimal)	float	2.34, -0.534, -18.4e-5
Strings (text)	str	“John”, ‘abcd’, ‘01234’

Example: Calculating the Area of a Circle

$$A = \pi r^2$$

```
>>> r = 5
>>> A = 3.14*r**2
>>> print(A)
78.5
```

R is an int type

A is a float type

Notation is different than in Math

We use '*' instead of 'x' for multiplication

We indicate exponentiation with ''**

Variables

- Variables in action

```
>>> r = 5
>>> A = 3.14*r**2
>>> print(A)
78.5
```

r and A are variables. These variables are different from the variables we learned in Algebra.

Variables

```
>>> r = 5
>>> A = 3.14*r**2
```

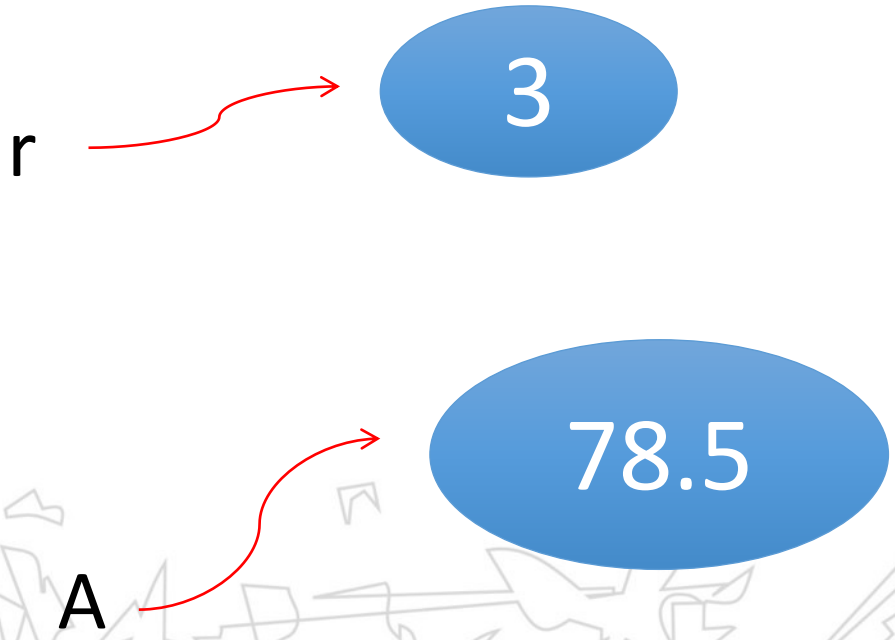
The value of ‘r’ is 5
The value of A is 78.5

r → 5 **A → 78.5**

A variable is a named memory location. Think of a variable as a box.

It contains a value(object). Think of the value as the contents of the box

Reference Diagram



Assignment Statements

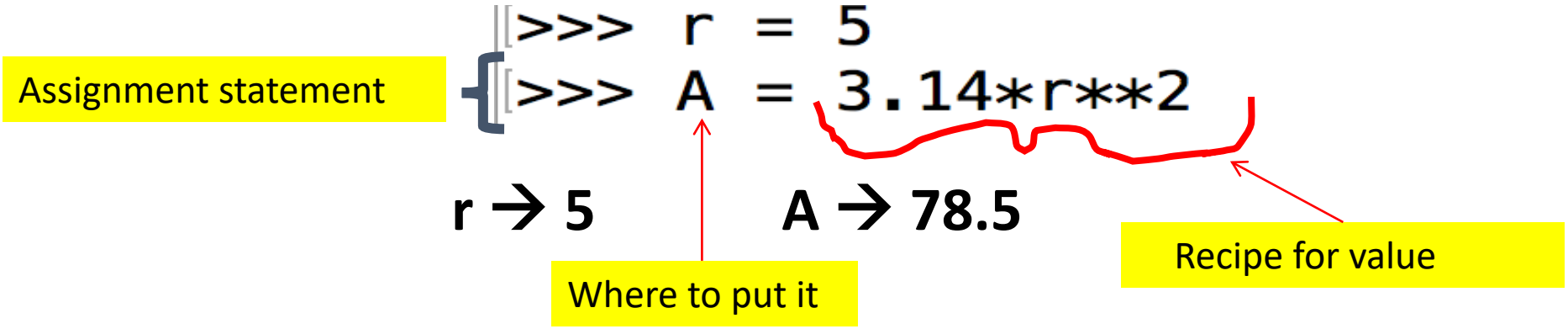
```
>>> r = 5
```

'r' is assigned to the value of 5
'r' gets the value 5

r → 5

The '=' token indicates assignment. The assignment statement r = 5 creates variable r and assigns it to the value of 5.

Assignment Statements



A variable can be used in an expression like: 3.14*r2**

The expression is evaluated and then stored

Order of Operations is Important

```
>>> A = 3.14*r**2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'r' is not defined
>>>
```

Need to declare variable first in an earlier statement, before you can use on the right-hand side

Assignment vs. 'Equal to'

```
>>> r = 5
>>> 3.14*r**2 = A
  File "<stdin>", line 1
SyntaxError: can't assign to operator
>>>
```

- In math '=' → what is on the left is equal to what is on the right

- In python '=' → prescribes an action: evaluate whatever is on the right (rhs) and assign its value to the variable named on the left(lhs)
- To ask if both the rhs and lhs are equal in python we use **==**

>>> 7 == 3 + 4
True

>>> 8 == 3 + 4
False

Objects and Variables

- Names are not permanently assigned to objects. A name can be reassigned to another object, and multiple names can refer to the same object.

```
>>> r = 5
>>> A = 3.14*r**2
>>> A = A/2
```

r → ?
A → ?
A → ?

A *has been overwritten* by A/2

Assignments vs. Equations

- In Algebra,

$$t = t + 10$$

  - Doesn't make sense!

- In python,

$$t = t + 10$$

  - Means add 10 to the current value of 't' and store the result in 't', or increment 't' by 10.

Key 2-step process being every assignment statement

- <variable name> = < Evaluate me(expression) first>

1. Evaluate the expression on the right
2. Store the resulting value in the variable named on the left

Naming Variables

- Rules:

1. Names must be comprised of a combination of upper case letters, lower case letters, digits, or the underscore '_'
2. The first character in the name must be a letter or underscore

- Good Practice: Give names that are short and intuitive

```
>>> radius = 5
>>> Area = 3.14*radius**2
>>> Area
78.5
```

Integers and Decimals

- In math we distinguish between integer numbers and decimal numbers.

- Integer numbers: 100, -2, 20000, 3, 1
- Decimal numbers: 3.14, 0.345, -1.234,

Integers and Decimals

- There are different kinds of division

- Integer Division:

  30//8 => 3 with a remainder of 6

- Decimal Division:

  30/8 => 3.75

int vs float

- In python, a number, as well as any other object, has a type.
- The int type represents numbers as integers
- The float type represents numbers as decimals

It's important to understand the differences and the interactions

int vs float

```
>>> x = 30
>>> y = 8.
>>> print(type(x))
<class 'int'>
>>> print(type(y))
<class 'float'>
```

The python interpreter knows ‘x’ is of type int because it has no decimals. It also knows ‘y’ is of type float because it has a decimal point.

Arithmetic In Python

- In python 3 the '/' token performs decimal division

```
>>> x = 30
>>> y = 8
>>> q = x/y
>>> print(q)
3.75
```

- If you need integer division use the '//' token

```
>>> x = 30
>>> y = 8
>>> q = x//y
>>> print(q)
3
```

Order of Precedence

1. Exponentiation and negation
2. Multiplication and division
3. Addition and subtraction

Operator	Associativity
** (exponentiation)	right-to-left
- (negation)	left-to-right
* (mult), / (div), // (truncating div), % (modulo)	left-to-right
+ (addition), - (subtraction)	left-to-right

- So, these statements are equivalent:

A + B * C == A + (B*C)

A*B**2+C == (A*(B**2))+C

It's a good practice to use parenthesis to avoid ambiguity

String (str) Data Type

- So far we have discussed computations with numbers
- Now, let's talk about computation with text or Strings
- We use the type str to manipulate text in python

String (str) Data Type

- Strings (str) in python are quoted characters

```
>>> s1 = 'my string'
>>> s2 = "This is another string example"
>>> s3 = """So is this!"""
>>> s4 = '''   And this one '''
```

String (str) Data Type

```
>>> s1 = 'abc'
>>> s2 = 'ABC'
>>> s3 = ' A B C'
```

- The values in s1, s2 and s3 are all different. Capitalization and spaces matter.

- There's nothing special about letters, digits, punctuation and special characters can also be values for str type. Although some rules apply to special characters that are part of python's syntax.

Types

Type	Python Type	Examples
Integers	int	123, -123, 0
Real valued (decimal)	float	2.34, -0.534, -18.4e-5
Strings (text)	str	“John”, ‘abcd’, ‘01234’

A Type Is A Set Of Values And Operations On Them

- Values …

- int 123, -56, 7890000
- float 0.567, -0.04, -13e-5
- str “Julia’s choice”, ‘1234’, ‘abc’

These are also called literals, i.e., no variable to store/associate them

A Type Is A Set Of Values And Operations On Them

- Operations …

- int + * // / unary- % **
- float + * // / unary- **
- str + *

Duplicate string

concatenation

Formatted Strings From Python 3.6+

- f-strings
  - a cool feature introduced in python 3.6

```
name = "Eva"
age = 44
print(f"The principal said her name was {name}. She is {age} years old.")
'The principal said her name is Eva. She is 44 years old.'
```

  - Lets you avoid the need for concatenation and casting in one go… pretty cool stuff

Formatted Strings From Python 3.6+

- f-strings
  - modifiers can be used to convert the value before it is formatted. '!a' applies ascii(), '!s' applies str(), and '!r' applies repr() or printable representation:

```
>>> animals = 'eels'
>>> print(f'My hovercraft is full of {animals}.')
My hovercraft is full of eels.
>>> print(f'My hovercraft is full of {animals!r}.')
My hovercraft is full of 'eels'.
```

  - But also lets you do conversion/casting if necessary.

Explicit Type Conversion

```
>>> s1 = '123.45'
>>> x = 2*float(s1)
>>> print(x)
246.9
```

- A string that encodes a valid decimal number can be converted to a float using the built-in function float()

Explicit Type Conversion

```
>>> s1 = '123'
>>> x = 2*int(s1)
>>> print(x)
246
```

- A string that encodes a valid integer number can be converted to a **int** using the built-in function **int()**

Automatic Type Conversion

```
>>> x = 1/2
>>> print(x)
0.5
>>> y = 2*x
>>> print(y)
1.0
```

- An operation between an int and a float results in a float. So ‘y’ ends up being a float even though its value happens to be an integer.

Python Is A Dynamically Typed Language

```
>>> x = 23.4
>>> x = 'some text'
>>> x = 10000
```

- A variable can hold different data types at different points in time.

Summary

- Variables hold values that can be accessed by our programs
- Assignment statements assign values to variables
- Numerical data can be represented in python using the int and float data types
- Text can be represented in python using the str data type.

Input

- Program examples we've seen don't take any input from the user
- Python built in function input() gets input from the keyboard
- How does input() works?
- When input() is called, the program stops to receive user input
- When the user pressed Return or Enter, the program resumes and the input() returns the data entered as a string.

Input Example

- It is always a good idea to print a prompt telling the user what to input as in the example below.

```
name = input ("What is your name?\n")
What is your name?
name
'Jones'
```

The '\n' is the new line sequence. It causes a line break.

Input Example

- If you require a different data type, you can convert the return value.

```
gpa = input ("What is your GPA?\n")
What is your gpa?
type (gpa)
<class 'string'>
gpa= float(gpa)
type (gpa)
<class 'float'>
```

The input received must be a valid sequence to be converted. Otherwise you will get an error message.

Input As Command Line Arguments

- Remember from Linux or command line classes you can provide options at command line.

program.exe input_or_option1 input_or_option2 ….

- Your program is able to use those inputs

- In python scripting you can provide same types of inputs with using the argument object “argv” from the system library.

python add.py input_or_option1 input_or_option2

Input As Command Line Arguments

add.py contents:

```
import sys
a= sys.argv[1]
b= sys.argv[2]

print ("sum of", a, "and", b , "is",   float (a) + float (b)  )
```

In Class Exercise: Input As Command Line Arguments

- Modify add.py contents:
  - Use f-string to rewrite
    - print ("sum of", a, "and", b , "is", float (a) + float (b))
  -

```
import sys
a= sys.argv[1]
b= sys.argv[2]

print (f" ???? ")
```

Type Conversion (or Casting)

- `type(name):` to learn the type of a variable

- Python does some automatic (or implicit) conversion (e.g between **int** and **float**)

- Explicit conversions
- `int()` converts from float or strings of integers
- `float()` converts from int or strings
- `str()` converts from anything

Output

- We have been using the print() function to send data to the terminal or screen.
- We can print any number of values separated by commas

```
print("There are", 34*60, "seconds in 34 minutes")
There are 2040 seconds in 34 minutes
```

Variables and expressions given as arguments to the print function are evaluated before they're send to the screen.

Summary

- We can use the input() function in python to request input from the user
- Data received from input() is of the type 'str'
- We send output to the screen using the print() function.