IMPULSANDO EL CONOCIMIENTO
TIC CUALIFICADO

EDUCACIÓN
CALIDAD Y CERTIFICACIÓN
EN SUS PROGRAMAS EDUCATIVOS

Programación de Redes – Becas Digitaliza - 2019

PUE – ITC – Formación de Instructores

Sesión 2 – Introducción a Python (I)

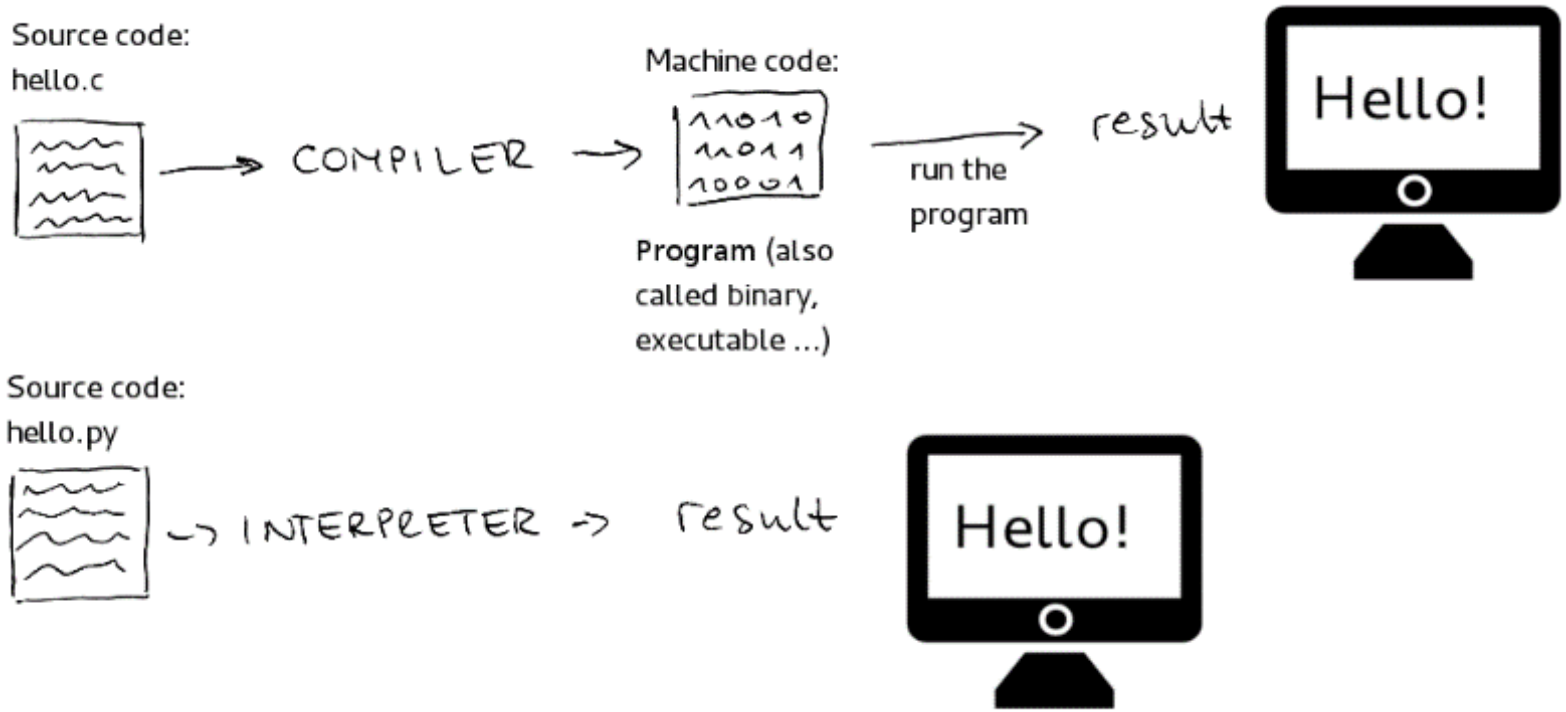
Iván Lago - Técnico Cisco Networking Academy ASC/ITC
PUE - ITC/ASC/CA

Index

- Python basics
  - Compiled vs Interpreted: which is Python?
  - Data types
  - Conversions (casting)
  - Decisions
  - List, dictionaries
  - Loops

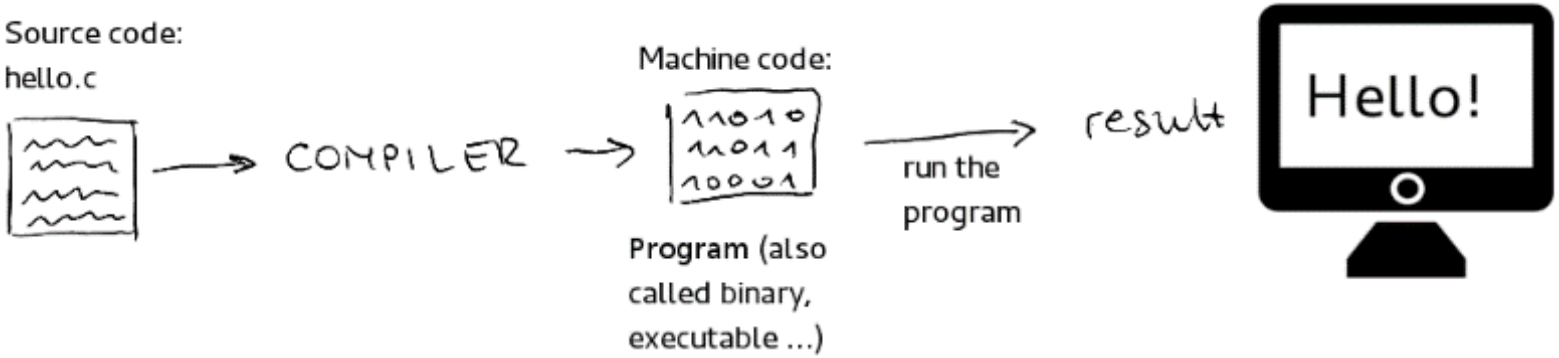
Python Basics

Compiled vs Interpreted (I)



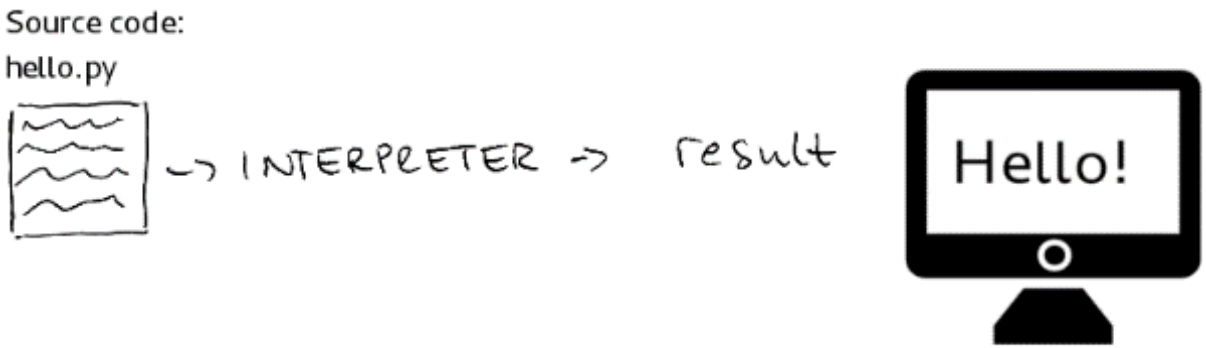
Compiled vs Interpreted (II)

- **Compiled:** the program, once compiled, is expressed in the instructions of the target machine. For example, an addition "+" operation in your source code could be translated directly to the "ADD" instruction in machine code.
  - Faster performance by directly using the native code of the target machine
  - Opportunity to apply quite powerful optimisations during the compile stage



Compiled vs Interpreted (III)

- **Interpreted:** the instructions are not directly executed by the target machine, but instead read and executed by some other program (which normally is written in the language of the native machine). For example, the same "+" operation would be recognised by the interpreter at run time, which would then call its own "add(a,b)" function with the appropriate arguments, which would then execute the machine code "ADD" instruction.
  - Easier to implement (writing good compilers is very hard!!)
  - No need to run a compilation stage: can execute code directly "on the fly"
  - Can be more convenient for dynamic languages



Start Python: Interpreter

Windows

```
C:\> python
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Mac or Linux

```
$ python3
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use Interactive Interpreter as a Calculator

```
$ python3
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> 10-4
6
>>> 2*4
8
>>> 20/5
4
>>> 3**2
9
```

Use Interpreter to print Hello World

- Strings can be enclosed with single quotes or double quotes.
- To remove the single quotes in the output, use the print command.

```
>>> "Hello World!"
'Hello World!'
>>> 'Hello World!'
'Hello World!'
>>> print("Hello World!")
Hello World!
```

Quit the Interpreter and Start IDLE

- Python includes the Integrated Development Environment (IDLE)
- Windows - open IDLE from the Start menu
- Mac or Linux - open IDLE from the command line.

Windows

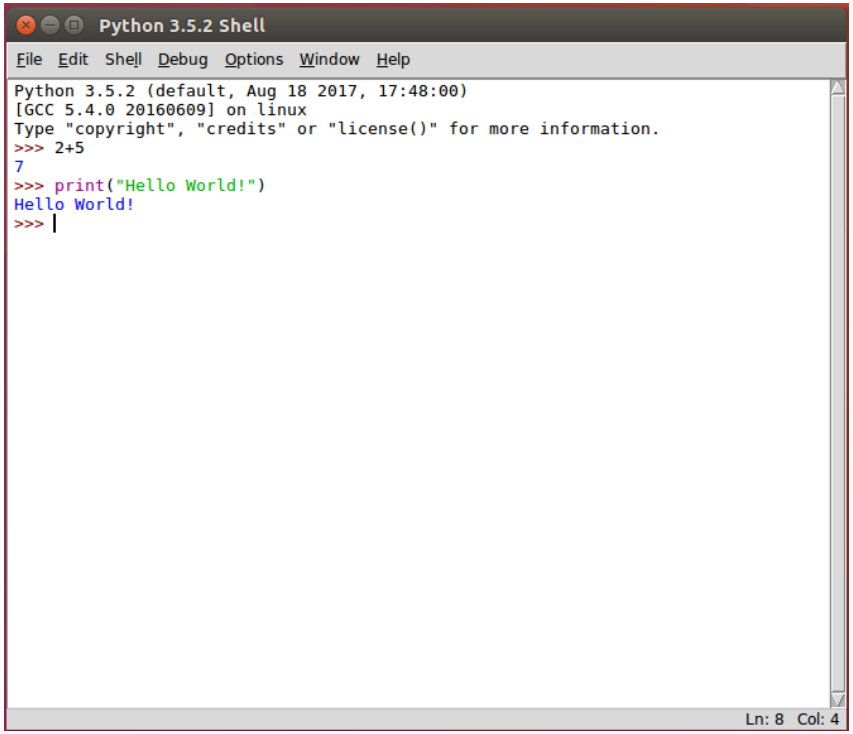
Start > Python 3.6 > IDLE (Python 3.6 32-bit).

Mac or Linux

```
>>> "Hello World!"
'Hello World!'
>>> 'Hello World!'
'Hello World!'
>>> quit()
$ idle3
```

IDLE Benefits

- Integrated development and learning environment
- Provides color coding
- Includes a text editor for writing programs
- Quickly save and run programs



IDE (I): Jupyter and PyCharm

```
In [ ]: # First Let's setup the connection and auth:

        from azureml import Workspace

        ws = Workspace(workspace_id='34d9dbb5ab8043b0b432a4c02f37ee0e',
                       authorization_token='89c99cd805ba4920b95e99aa9513d133',
                       endpoint='https://studioapi.azureml-int.net' )
```

To enumerate all the example Experiments:

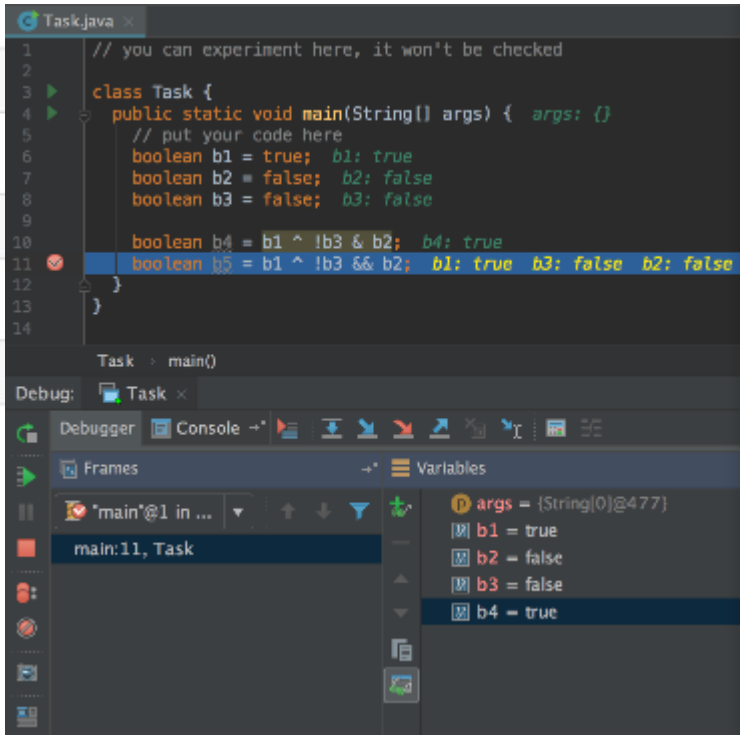
```
In [ ]: for ex in ws.example_experiments:
            print (ex.description)
```

To view all the datasets:

```
In [ ]: for ds in ws.datasets:
            print(ds.name)
```

Or just the user-created datasets:

```
In [ ]: for ds in ws.user_datasets:
            print(ds.name)
```



Help, hints, console, debugger, projects…

IDE (II): Eclipse

Debug - IPX v2/resources/libs/sources/PassionXbmc/PassionXbmcItemInstaller.py - Eclipse SDK

File Edit Source Refactoring Navigate Search Project Pydev Tomcat Run Window Help

Debug Server [Python Server]
unknown
pydevd.reader - -1
pydevd.writer - -1
MainThread - pid1296_seq1
_downloadFile [PassionXbmcItemInstaller.py:168]
downloadItem [PassionXbmcItemInstaller.py:61]
installItem [ItemInstaller.py:389]
install_add_ons [MainGui.py:569]
onClick [MainGui.py:681]
show_main [MainGui.py:1186]
show_main [Home.py:197]

"item"= NameError: name 'item' is not defined
"headers"= NameError: name 'headers' is not defined
"self.filename"= unicode: Calendar.rar
"self.filesize"= int: 5272496
"self.url"= unicode: http://passion-xbmc.org/downloa

```
154                         try:
155                             self.filesize = int( headers['Content-Length'] )
156                         except:
157                             self.filesize = 0

159                 except Exception, e:
160                     self.filename = "unknownfilename"
161                     status = "ERRORFILENAME"
162                     self.filesize = 0
163                     print("_downloadFile - Exception retrieving header")
164                     print(str(e))
165                     print_exc()
166
```

os
sys
httplib
urllib2
print_exc (traceback)
xbmc
CONF
* (utilities)
ArchItemInstaller.cancelRequest (ItemInstaller)
httplib.HTTPConnection.debuglevel
_
PassionXbmcItemInstaller
__init__

```
[SCRIPT IPX]: self.destinationPath
[SCRIPT IPX]: C:\Working\XBMC\Dev\XBMC\scripts\Calendar
[SCRIPT IPX]: self.installName
[SCRIPT IPX]: Calendar
[SCRIPT IPX]: installItem - Item is already installed - stopping install
[SCRIPT IPX]: Calendar install has been cancelled by the user
[SCRIPT IPX]: {'xbmc_type': 'SCRIPT', 'added': '24-02-2010', 'image2retrieve': True, 'name': u'Calendar', 'parent': 12, 'author': u'', 'previewpicture': '',
[SCRIPT IPX]: _downloadFile with destination directory = C:\Working\XBMC\Dev\XBMC\userdata\script_data\Installer Passion-XBMC\cache
[SCRIPT IPX]: _downloadFile - Trying to retrieve the file
```

Activity - Write, Save, and Run Your First Program

1. In IDLE, click **File > New File (Ctrl+N)** to open an Untitled script file.
2. Save the file as 01_hello-world.py.
3. Enter the following in the script:

   print("Hello World!")
4. Save the script; click **File > Save (Ctrl+S)**
5. Run the script; click **Run > Run Module (F5)**

First Program and Output

01_Hello-World.py - /home/allan/Dropbox/ETW-BP/01_Hello-W

File Edit Format Run Options Window Help

```
print("Hello World!")
```

Ln: 2 Col: 0

Program File

Python 3.5.2 Shell

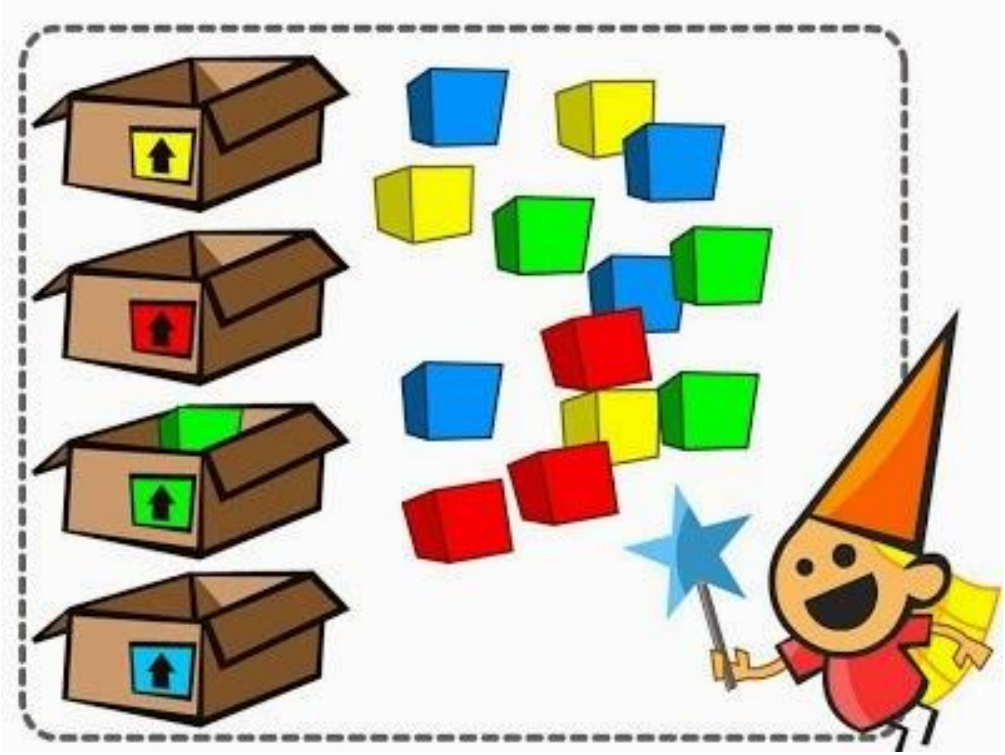
File Edit Shell Debug Options Window Help

```
Python 3.5.2 (default, Aug 18 2017, 17:48:00)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+5
7
>>> print("Hello World!")
Hello World!
>>>
=========== RESTART: /home/allan/Dropbox/ETW-BP/01_Hello-World.py
===========
Hello World!
>>>
```

Ln: 11 Col: 4

Output

Data Types: classification (I)



Data Types: classification (II)

Data Types: classification (III)

1 2 3 4 5
6 7 8 9 10
1 2 3 4 5
6 7 8 9 10

121.34

A B C D E
F G H I J K
L M N O P
Q R S T U
V W X Y Z

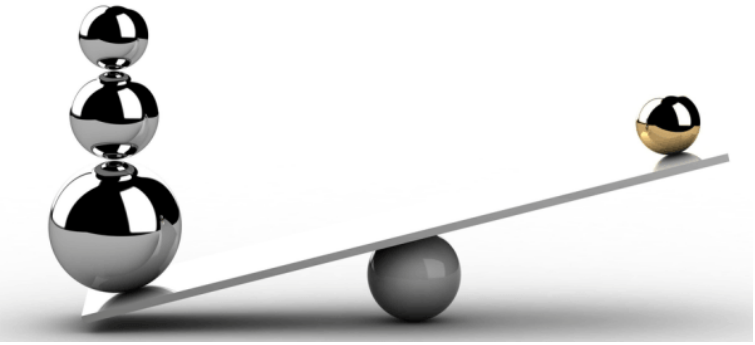
Basic Data Types

- The four basic data types we will use are:
  - Integer
  - Float
  - String
  - Boolean

- Use the type() command to determine the data type.

```
>>> type(98)
<class 'int'>
>>> type(98.6)
<class 'float'>
>>> type("Hi!")
<class 'str'>
>>> type(True)
<class 'bool'>
```

Boolean Comparison Operators



Operator	Meaning
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

```
>>> 1<2
True
>>> 1>2
False
>>> 1==1
True
>>> 1!=1
False
>>> 1>=1
True
>>> 1<=1
True
```

Creating and Using a Variable

- Use a single equal sign to assign a value to a variable.
- A variable can then be called for other operations.

```
>>> x=3
>>> x*5
15
>>> "Cisco"*x
'CiscoCiscoCisco'
```

Concatenate Multiple String Variables

Concatenation is the process of combining multiple strings.

```
>>> str1="Cisco"
>>> str2="Networking"
>>> str3="Academy"
>>> space=" "
>>> print(str1+space+str2+space+str3)
Cisco Networking Academy
>>>
```

Converting Data Types

- Concatenation does not work for different data types.

```
>>> x=3
>>> print("This value of X is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str
implicitly
```

Converting Data Types

Use the **str()** command to convert the data type to a string.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str
implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>>
```

Converting Data Types

The type for the variable x is still an integer.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str
implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
```

Converting Data Types: casting

To convert the data type, reassign the variable to the new data type.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    print("This value of X is " + x)
TypeError: Can't convert 'int' object to str
implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
>>> x=str(x)
>>> type(x)
<class 'str'>
```

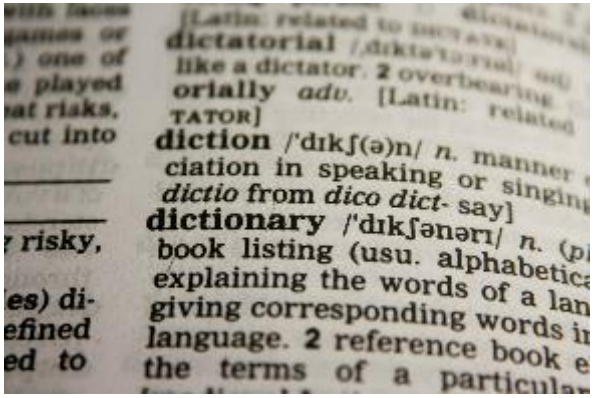
Converting Data Types: decimal

- Use **“{:.2f}”.format** to display a float to two decimal places.

- Change the **2** to increase or decrease decimal places.

```
>>> pi = 22/7
>>> print(pi)
3.142857142857143
>>> print("{:.2f}".format(pi))
3.14
>>>
```

Lists and Dictionaries



Lists (I)

- A list is an ordered list of items.
  - Create a list using the brackets **[ ]** and enclosing each item in the list with quotes.
  - Use the **type()** command to verify the data type.
  - Use the **len()** command return the number of items in a list.
  - Call the list variable name to display it's contents.

```
>>> hostnames=["R1","R2","R3","S1","S2"]
>>> type(hostnames)
<class 'list'>
>>> len(hostnames)
5
>>> hostnames
['R1', 'R2', 'R3', 'S1', 'S2']
```

Lists (II)

- Use the index to refer to an item and manipulate the list
  - The first item in a list is indexed as zero, the second is indexed as one, and so on.
  - The last item can be referenced with index **[-1]**
  - Replace an item by assigning a new value to the index.
  - Use the **del()** command to remove an item from a list.

```
>>> hostnames=["R1","R2","R3","S1","S2"]
>>> type(hostnames)
<class 'list'>
>>> len(hostnames)
5
>>> hostnames
['R1', 'R2', 'R3', 'S1', 'S2']
>>> hostnames[0]
'R1'
>>> hostnames[-1]
'S2'
>>> hostnames[0]="RTR1"
>>> hostnames
['RTR1', 'R2', 'R3', 'S1', 'S2']
>>> del hostnames[3]
>>> hostnames
['RTR1', 'R2', 'R3', 'S2']
>>>
```

Dictionaries (I)

- A list of unordered key/value pairs
  - Create a dictionary using the braces **{ }**
  - Each dictionary entry includes a key and a value.
  - Separate key and values with a colon.
  - Use quotes for keys and values that are strings.

```
>>> ipAddress =
{"R1":"10.1.1.1","R2":"10.2.2.1","R3":"10.3.3.1"}
>>> type(ipAddress)
<class 'dict'>
```

Dictionaries (II)

- Use the key to refer to an entry
  - The key is enclosed with brackets **[ ]**.
  - Keys that are strings can be referenced using single or double quotes.
  - Add a key/value pair by setting the new key equal to a value.
  - Use ***key* in *dictionary*** command to verify if a key exist in the dictionary

```
>>> ipAddress =
{"R1":"10.1.1.1","R2":"10.2.2.1","R3":
"10.3.3.1"}
>>> type(ipAddress)
<class 'dict'>
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1',
'R3': '10.3.3.1'}
>>> ipAddress['R1']
'10.1.1.1'
>>> ipAddress["S1"]="10.1.1.10"
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1',
'R3': '10.3.3.1', 'S1': '10.1.1.10'}
>>> "R3" in ipAddress
True
>>>
```

Dictionaries (III)

- Lists key+value -> dictionary.items()
- List keys -> dictionary.keys()
- List values -> dictionary.values()

- for a,b in dictionary.items():
  print(b)
  …

- Loops + Dictionaries + Lists + Conditionals…

- for a,b in dictionary.items():
  if “something” in b:
    print(“We got it”)
    do something

Activity - Troubleshoot List and Dictionary Code

1. Open 02_list-dicts.py.
2. Run the code.
3. Troubleshoot the code until the script runs without errors.
4. What errors did you fix in the script?

User Input

JUST GIVE ME ALL THE DATA!

The Input Function

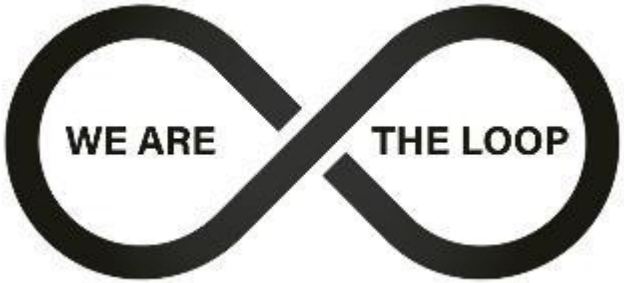
The **input()** function provides a way to get information from the user.

```
>>> firstName = input("What is your first name? ")
What is your first name? Bob
>>> print("Hello " + firstName +"!")
Hello Bob!
>>>
```

Activity - Create a Script to Collect Personal Information

1. Open a blank script file and named it as 03_personal-info.py.
2. Create a script that asks for four pieces of information such as: first name, last name, location, and age.
3. Create a variable for a space: **space = " "**
4. Add a print statement that that combines all the information in one sentence.
5. Run the script and troubleshoot any errors.

If functions and loops



If/Else Function (II)

- How to take decisions
- Think about your life day
- Conditions are true?

```
if TheWeatherIsGood:
   GoForAWalk()
elif TicketsAvailable:
   GoToATheatre()
elif TableAvailable:
   GoForLunch()
else:
   PlayChessAtHome()
```

If/Else Function (II)

- Open a blank script and save it as 04_if-vlan.py.
- Create a simple **if** function that compares two values and prints the results.
- Run the script and troubleshoot any errors.
- Change the values to test the **else** print statement.

```
nativeVLAN = 1
dataVLAN = 100
if nativeVLAN == dataVLAN:
    print("The native VLAN and
the data VLAN are the same.")
else:
    print("This native VLAN and
the data VLAN are different.")
```

If/Elif/Else Function

- Open a blank script and save it as 05_if-acl.py.
- Create a more complex **if** function that takes user input and includes an **elif** loop.
- Note that the input needs to be converted to an integer.

```
aclNum = int(input("What is the IPv4
ACL number? "))
if aclNum >= 1 and aclNum <= 99:
    print("This is a standard IPv4
ACL.")
elif aclNum >=100 and aclNum <= 199:
    print("This is a extended IPv4
ACL.")
else:
    print("This is not a standard or
extended IPv4 ACL.")
```

For Loop

- A for loop iterates through items in a list, dictionary, or other sequenced data type.

```
>>>
devices=["R1","R2","R3","S1","S2"]
>>> for item in devices:
        print(item)


R1
R2
…
```

- The variable name “item” is arbitrary and can be anything the programmer chooses.

- Range of numbers

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

```
for i in range(10):
    print("The variable is: "+str(i))
```

For Loop with Embedded If

Using an If loop inside the For loop

```
>>> for item in devices:
        if "R" in item:
                print(item)


R1
R2
R3
>>>
```

Use a For Loop to Create a New List

- Create an empty list called switches.
- Iterate through the devices list to create the switch list.

```
>>> switches=[]
>>> for item in devices:
        if "S" in item:
                switches.append(item)


>>> switches
['S1', 'S2']
>>>
```

While Loop

- Do something “while” a condition is true

```
counter = 5;
while counter != 0:
   print("My name is Python...")
   counter -= 1
```

- We can break the loop

```
while True:
    a+=1
    print("Hello")
    if a==5:
        break
```

Create a While Loop

- Open a blank script and save it as 06_while-loop.py.

- Create a program with a while loop that counts to a user's supplied number.
  - Convert the string to an integer: **x = int(x)**
  - Set a variable to start the count: **y = 1**
  - **While y <= x**, print the value of y and increment y by 1.

```
x=input("Enter a number to
count to: ")
x=int(x)
y=1
while y<=x:
    print(y)
    y=y+1
```

Modify the While Loop to Use Break

Modify the while loop to use a Boolean check and break to stop the loop.

- Replace **while y<=x** with **while True**
- Add an if function to break the loop when **y>x**.

```
x=input("Enter a number to
count to: ")
x=int(x)
y=1
while True:
    print(y)
    y=y+1
    if y>x:
        break
```

Use a While Loop to Check for User Quit

- Add another while loop to the beginning of the script which will check for a quit command.

- Add an if function to the while loop to check for ‘q’ or ‘quit’ .

```
while True:
    x=input("Enter a number to
count to: ")
    if x == 'q' or x == 'quit':
        break
x=int(x)
y=1
while True:
    print(y)
    y=y+1
    if y>x:
        break
```

Gracias por vuestra atención

Iván Lago - Técnico Cisco Networking Academy ASC/ITC
PUE - ITC/ASC/CA
Área de Proyectos de Educación