Python for DevNet Associates

Patrick Rockholz
parockho@cisco.com
Systems Architect

Brian Buxton
brbuxton@cisco.com
Systems Architect

Agenda

- Overview
- Code Editor vs IDE
- Libraries, Pip, virtualenv
- Data types
- If Statements
- Loop Statements
- Functions
- Code Testing
- Challenge Winners & Kahoot

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

python

Python is powerful... and fast;

plays well with others;

runs everywhere;

is friendly & easy to learn;

is Open.

https://www.python.org/about/

cisco

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Why Python for IT Engineers?

- Python is a good starting point. (one of the easiest)
  - it can be interpreted like a script
  - it is simple and very readable.
- Python is a simple language.
  - was built so that common tasks are easy to accomplish.
- Python has a large community of network-based code
  - many of the tasks we will want to accomplish have already been coded.
- Python is extensible.
  - Whether through libraries and packages, or through the more advanced features of the language itself, Python moves easily from simple tasks to more involved and advanced ones.

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

How to Run Python

Interactive Interpreter

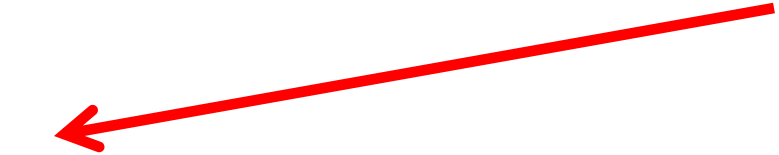
```
PAROCKHO-M-R0B2:~ parockho$ python3
Python 3.6.4 (default, Jan  6 2018, 11:51:59)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)] on
darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> exit()
PAROCKHO-M-R0B2:~ parockho$
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

How to Run Python

Scripts

```
PAROCKHO-M-R0B2:~ parockho$ python3 example1-1.py
```



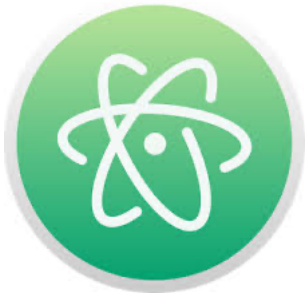
```
def doubler(number):
    result = number * 2
    return result
get_number = float(input("Type a number: "))
answer = doubler(get_number)
print(answer)
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

How to Run Python

Code Editor = a text editor with some syntax highlighting and code formatting capabilities

(e.g. vi/vim, NotePad++, Atom, BBEdit)



https://realpython.com/python-ides-code-editors-guide/

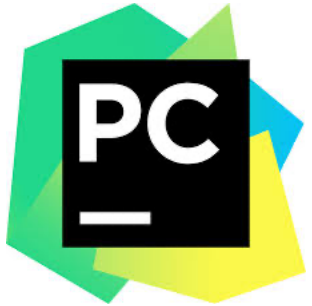
© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

How to Run Python

IDE (Integrated Development Enviornment)

- An editor designed to handle code (with, for example, syntax highlighting and auto-completion)
- Build, execution, and debugging tools
- Some form of source control

(e.g. PyCharm, Visual Studio Code)



https://realpython.com/python-ides-code-editors-guide/

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Demo Time!

How to Run Python

A Good Coding Enviornment

- The ability to save your code and pick up right where you let off
- Run your code within the code enviornment
- The capability to “step through” the code for troubleshooting and debugging
- Syntax highlighting readability and understanding
- Code formatting assistance (e.g. colons, indentations, etc)

https://realpython.com/python-ides-code-editors-guide/

cisco

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Python Libraries (Modules, Applications, etc)

- Any Python code outside of your script you want to use
- Provide some capability or data you need
- Included with statements
  - `from` **`library`** `import` **`name`**
  - `import` **`library`**

```
# Import data from another script
from common_vars import shapes
# Import a library that offers date-time capabilities
import datetime

print("The shapes are:")
for shape in shapes:
    print(shape)
print("")

# Get Current Date and Time
date_now = datetime.datetime.now()
print("It is currently {}.".format(str(date_now)))

# Add 1000 minutes to Current Date and Time
new_date = date_now + datetime.timedelta(minutes=1000)
print("In 1000 minutes it will be {}.".format(str(new_date)))
```

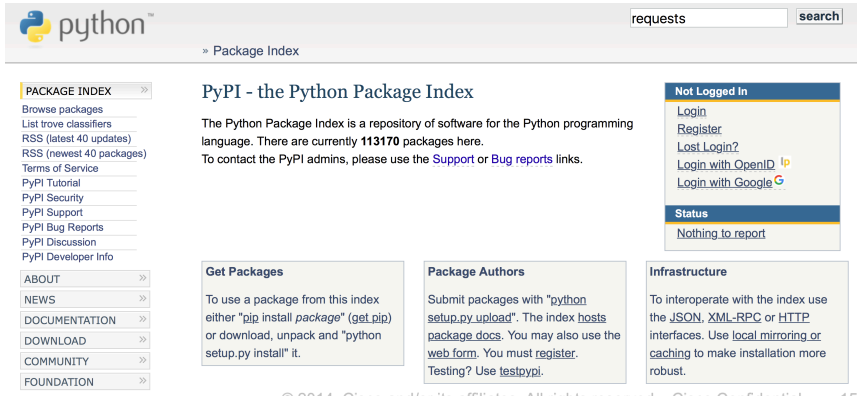

netprog_basics/programming_fundamentals/python_part_2/example1.py

Where to get Libraries

- Write them yourself
  - Example: `common_vars`
- Included with Python itself
  - Some must be imported (print vs pprint)
  - Example: `datetime, os, sys, json`
- From Python Package Index (PyPI)
  - Example: `pip install requests`
- Download and install manually
  - Example: ACI Toolkit from GitHub

```
#! /usr/bin/env python
""" ... """

shapes = ["square", "triangle", "circle"]
books = [
            {
                "title": "War and Peace",
                "shelf": 3,
                "available": True
            },
            {
                "title": "Hamlet",
                "shelf": 1,
                "available": False
            },
            {
                "title": "Harold and the Purple Crayon",
                "shelf": 2,
                "available": True
            }
        ]
colors = ["blue", "green", "red"]
```



© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_2/common_vars.py

Pip, The Python Package Installer

- Python 2.7.6 and Python 3.4 or greater includes by default
- Integrates with PyPI for packages
- Install, Upgrade, and Uninstall packages
- "`requirements.txt`" in projects provide input to pip

```
DevNet$ pip --version
pip 9.0.1

DevNet$ pip install pyang
Collecting pyang
  Downloading pyang-1.7.3-py2.py3-none-any.whl (326kB)
    100% |████████████████████████████████| 327kB 1.3MB/s
Installing collected packages: pyang
Successfully installed pyang-1.7.3

DevNet$ pip install -r requirements.txt
Collecting requests (from -r requirements.txt (line 1))
  Downloading requests-2.18.2-py2.py3-none-any.whl (88kB)
    100% |████████████████████████████████| 92kB 662kB/s
.
{OUTPUT TRUNCATED}
.
Successfully installed requests-2.18.2 six-1.10.0 urllib3-
1.22
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_2/requirements.txt

pip (and pip3) Commands to Know

- Install a package
  - `pip install` **`package`**
- Upgrade a package
  - `pip install --upgrade` **`package`**
- Uninstall a package
  - `pip uninstall` **`package`**
- View all packages installed
  - `pip freeze`
- Install "`requirements.txt`"
  - `pip install -r requirements.txt`

```
DevNet$ pip freeze

asn1crypto==0.22.0
bcrypt==3.1.3
certifi==2017.4.17
cffi==1.10.0
chardet==3.0.4
cryptography==2.0
flake8==3.3.0
idna==2.5
lxml==3.8.0
mccabe==0.6.1
ncclient==0.5.3
paramiko==2.2.1
pyang==1.7.3
pyasn1==0.2.3
pycodestyle==2.3.1
pycparser==2.18
pyflakes==1.5.0
PyNaCl==1.1.2
requests==2.18.2
six==1.10.0
urllib3==1.22
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

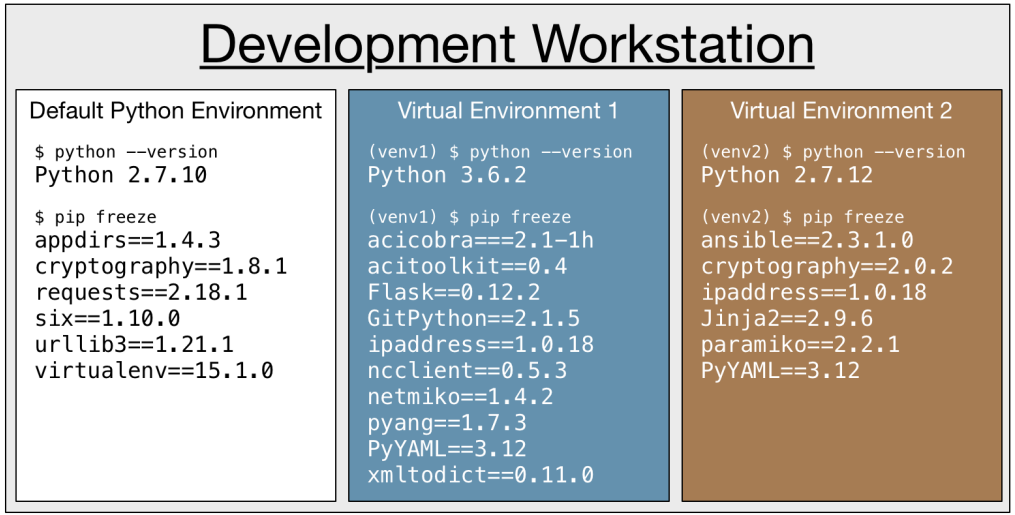
What is a Virtual Environment (venv)

1) Global Enviornment
   1) Used by all .py programs
   2) What if we want different Python versions for testing?
   3) What if we have different library dependencies?
2) Virtual Enviornment
   1) `python3 -m venv [directory name] e.g. python3 -m venv` **`name`**
   2) Activate – `source` **`name`**`/bin/activate`
   3) Once done, `deactivate`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

What is a Virtual Environment (venv)

- Build isolated, fully functional Python environments on a single workstation
- Virtual Environments can
  - Run different versions of Python
  - Have different libraries installed
  - Have different versions of libraries installed



© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Installing Python Libraries in Virtual Environments

- Once activated, no different

```
(venv) DevNet$ pip install pyang
Collecting pyang
  Downloading pyang-1.7.3-py2.py3-none-any.whl
(326kB)
    100% |████████████████████████████████| 327kB
1.3MB/s
Installing collected packages: pyang
Successfully installed pyang-1.7.3

(venv) DevNet$ pip install -r requirements.txt
Collecting requests (from -r requirements.txt (line
1))
  Downloading requests-2.18.2-py2.py3-none-any.whl
(88kB)
    100% |████████████████████████████████| 92kB
662kB/s
.
{OUTPUT TRUNCATED}
.
Successfully installed requests-2.18.2 six-1.10.0
urllib3-1.22
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Demo Time!

Core Python Libraries to Know and Love

- Pretty Print
  - `from pprint import pprint`
- Python Interpreter Utilities
  - `import sys`
- Operating System Interfaces
  - `import os`
- Date and Time Utilities
  - `import datetime`

* Many libraries included with core Python, see docs

cisco
© 2023 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Prettier Printing with pprint

- Better formatting than default print() function

```
>>> from pprint import pprint
>>> from common_vars import *
>>>
>>> print(books)
[{'title': 'War and Peace', 'shelf': 3, 'available': True}, {'title': 'Ha
mlet', 'shelf': 1, 'available': False}, {'title': 'Harold and the Purple
Crayon', 'shelf': 2, 'available': True}]
>>>
>>> pprint(books)
[{'available': True, 'shelf': 3, 'title': 'War and Peace'},
 {'available': False, 'shelf': 1, 'title': 'Hamlet'},
 {'available': True, 'shelf': 2, 'title': 'Harold and the Purple Crayon'}
]
>>>
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

https://docs.python.org/3/library/pprint.html

Access Details about Python with sys

- Access to some details/variables concerning running state
  - Access command line arguments with `sys.argv[]`
- Access to functions that interact with the interpreter
  - Exit Python with specific error message
    `sys.exit("Message")`

```
DevNet$ python -i common_vars.py "CLI Arg 1" "CLI Arg 2"
>>>
>>> import sys
>>>
>>> sys.argv[1]
'CLI Arg 1'
>>> sys.argv[2]
'CLI Arg 2'
>>>
>>> sys.exit("Error Occurred")
Error Occurred
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

https://docs.python.org/3/library/sys.html

Interact with Files, Paths, and Environment with os

- Access and manipulate directories and files
  - Note: Opening files can be done with `open(filename)`
- Access and Manipulate Environment Variables
  - `os.environ[var_name]`

```
>>> import os
>>> os.getcwd()
'/Users/hapresto/coding'
>>> os.chdir("../")
>>> os.getcwd()
'/Users/hapresto'
>>>
>>> os.environ["USER"]
'hapresto'
>>> os.environ["VAR_FROM_PYTHON"]
Traceback (most recent call last):
  File "<pyshell#46>", line 1, in <module>
    os.environ["VAR_FROM_PYTHON"]
  File "/Users/hapresto/coding/netprog_basics/venv/bin/
line 669, in __getitem__
    raise KeyError(key) from None
KeyError: 'VAR_FROM_PYTHON'
>>> os.environ["VAR_FROM_PYTHON"] = "Set from Python"
>>> os.environ["VAR_FROM_PYTHON"]
'Set from Python'
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

https://docs.python.org/3/library/os.html

Get your Date and Time Correct with datetime

- Create, format, and manipulate dates and times
- Time arithmetic!
- Work with timestamps and other representations

```
>>> import datetime

>>> right_now = datetime.datetime.now()

>>> four_weeks_from_now = right_now + datetime.timedelta(weeks=4)

>>> date_display_format = "%I:%m %p on %B %w, %Y"

>>> right_now.strftime(date_display_format)
'11:07 AM on July 3, 2017'
>>> four_weeks_from_now.strftime(date_display_format)
'11:08 AM on August 3, 2017'
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

https://docs.python.org/3/library/datetime.html

Manipulating Data of All Formats

- XML - xmltodict
  - `pip install xmltodict`
    `import xmltodict`
- JSON
  - `import json`
- YAML - PyYAML
  - `pip install PyYAML`
    `import yaml`
- CSV
  - `import csv`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Treat XML like Python Dictionaries with xmltodict

- Easily work with XML data
- Convert from XML -> Dict* and back
  - xmltodict.parse(**xml_data**)
  - xmltodict.unparse(**dict**)
- Python includes a native Markup (html/xml) interfaces as well
  - More powerful, but also more complex

https://pypi.python.org/pypi/xmltodict

* *Technically to an OrderedDict*

```
>>> import xmltodict
>>> from pprint import pprint
>>>
>>> xml_example = open("xml_example.xml").read()
>>> pprint(xml_example)
('<?xml version="1.0" encoding="UTF-8" ?>\n'
 '<interface xmlns="ietf-interfaces">\n'
 '  <name>GigabitEthernet2</name>\n'
 '  <description>Wide Area Network</description>\n'
 '  <enabled>true</enabled>\n'
 '  <ipv4>\n'
 '    <address>\n'
 '      <ip>172.16.0.2</ip>\n'
 '      <netmask>255.255.255.0</netmask>\n'
 '    </address>\n'
 '  </ipv4>\n'
 '</interface>\n')
>>>
>>> xml_dict = xmltodict.parse(xml_example)
>>> int_name = xml_dict["interface"]["name"]
>>> int_name
'GigabitEthernet2'
>>>
>>> xmltodict.unparse(xml_dict)
'<?xml version="1.0" encoding="utf-8"?>\n<interface xmlns="ietf-interf
gabitEthernet2</name><description>Wide Area Network</description><enab
led><ipv4><address><ip>172.16.0.2</ip><netmask>255.255.255.0</netmask>
v4></interface>'
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_3/xml_example.xml

To JSON and back again with json

- JSON and Python go together like peanut butter and jelly
  - `json.loads(`**`json_data`**`)`
  - `json.dumps(`**`object`**`)`
- JSON Objects convert to Dictionaries
- JSON Arrays convert to Lists

```
>>> import json
>>> from pprint import pprint
>>>
>>> json_example = open("json_example.json").read()
>>> pprint(json_example)
('{\n'
 '    "ietf-interfaces:interface": {\n'
 '        "name": "GigabitEthernet2",\n'
 '        "description": "Wide Area Network",\n'
 '        "enabled": true,\n'
 '        "ietf-ip:ipv4": {\n'
 '            "address": [\n'
 '                {\n'
 '                    "ip": "172.16.0.2",\n'
 '                    "netmask": "255.255.255.0"\n'
 '                }\n'
 '            ]\n'
 '        }\n'
 '    }\n'
 '}\n')
>>>
>>> json_python = json.loads(json_example)
>>> int_name = json_python["ietf-interfaces:interface"]["name"]
>>> int_name
'GigabitEthernet2'
>>>
>>> json.dumps(json_python)
'{"ietf-interfaces:interface": {"name": "GigabitEthernet2", "description":
a Network", "enabled": true, "ietf-ip:ipv4": {"address": [{"ip": "172.16.0
ask": "255.255.255.0"}]}}}'
```

https://docs.python.org/3/library/json.html

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_3/json_example.json

YAML? Yep, Python Can Do That Too!

- Easily convert a YAML file to a Python Object
  - `yaml.load(`**`yaml_data`**`)`
  - `yaml.dump(`**`object`**`)`
- YAML Objects become Dictionaries
- YAML Lists become Lists

```
>>> import yaml
>>> from pprint import pprint
>>>
>>> yml_example = open("yaml_example.yaml").read()
>>> pprint(yml_example)
('---\n'
 'ietf-interfaces:interface:\n'
 '  name: GigabitEthernet2\n'
 '  description: Wide Area Network\n'
 '  enabled: true\n'
 '  ietf-ip:ipv4:\n'
 '    address:\n'
 '    - ip: 172.16.0.2\n'
 '      netmask: 255.255.255.0\n')
>>>
>>> yaml_python = yaml.load(yml_example)
>>> int_name = yaml_python["ietf-interfaces:interface"]["name"]
>>> int_name
'GigabitEthernet2'
>>>
>>> yaml.dump(yaml_python)
'ietf-interfaces:interface:\n  description: Wide Area Network\n  en
tf-ip:ipv4:\n    address:\n    - {ip: 172.16.0.2, netmask: 255.255.
igabitEthernet2\n'
```

https://pypi.python.org/pypi/PyYAML/3.12

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_3/yaml_example.yaml

Import Spreadsheets and Data with csv

- Treat CSV data as lists
  - `csv.reader(file_object)`
- Efficiently processes large files without memory issues
- Options for header rows and different formats

```
>>> import csv
>>> import pprint
>>>
>>> csv_example = open("csv_example.csv").read()
>>> csv_example
'"router1","10.1.0.1","New York"\n"router2","10.2.0.1","Denver"\n"router3",
","Austin" \n'
>>>
>>> csv_example = open("csv_example.csv")
>>> csv_python = csv.reader(csv_example)
>>> for row in csv_python:
        print("{} is in {} and has IP {}".format(row[0], row[2], row[1]))


router1 is in New York and has IP 10.1.0.1
router2 is in Denver and has IP 10.2.0.1
router3 is in Austin  and has IP 10.3.0.1
```

https://docs.python.org/3/library/csv.html

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

netprog_basics/programming_fundamentals/python_part_3/csv_example.csv

Make HTTP Calls with Ease using requests

- Full HTTP Client
- Simplifies authentication, headers, and response tracking
- Great for REST API calls, or any HTTP request

```
>>> import requests
>>> from pprint import pprint
>>> router = {"ip": "10.10.20.21",
          "port": "443",
          "user": "root",
          "pass": "cisco123"}
>>>
>>> headers = {"Accept": "application/yang-data+json"}
>>>
>>> u = "https://{}:{}/restconf/data/interfaces/interface=GigabitEthernet1"
>>>
>>> u = u.format(router["ip"], router["port"])
>>>
>>> r = requests.get(u,
                     headers = headers,
                     auth=(router["user"], router["pass"]),
                     verify=False)
```

http://docs.python-requests.org

Make HTTP Calls with Ease using requests

- Full HTTP Client
- Simplifies authentication, headers, and response tracking
- Great for REST API calls, or any HTTP request

```
>>> pprint(r.text)
('{\n'
 '  "ietf-interfaces:interface": {\n'
 '    "name": "GigabitEthernet1",\n'
 '    "type": "iana-if-type:ethernetCsmacd",\n'
 '    "enabled": true,\n'
 '    "ietf-ip:ipv4": {\n'
 '      "address": [\n'
 '        {\n'
 '          "ip": "10.10.20.21",\n'
 '          "netmask": "255.255.255.0"\n'
 '        }\n'
 '      ]\n'
 '    },\n'
 '    "ietf-ip:ipv6": {\n'
 '    }\n'
 '  }\n'
 '}\n')
>>>
>>> api_data = r.json()
>>> interface_name = api_data["ietf-interfaces:interface"]["name"]
>>> interface_name
'GigabitEthernet1'
```

http://docs.python-requests.org

For When CLI is the Only Option – netmiko

- If no other API is available…
- Builds on paramiko library for SSH connectivity
- Support for a range of vendors network devices and operating systems
- Send and receive clear text
  - Post processing of data will be key

```
>>> from netmiko import ConnectHandler
>>> from pprint import pprint
>>>
>>> router = {"device_type": "cisco_ios",
          "host": "10.10.20.21",
          "user": "root",
          "pass": "cisco123"}
>>>
>>> net_connect = ConnectHandler(ip=router["host"],
                                 username=router["user"],
                                 password=router["pass"],
                                 device_type=router["device_type"])
>>>
>>> interface_cli = net_connect.send_command("show run int Gig1")
>>>
>>> pprint(interface_cli)
('Building configuration...\n'
 '\n'
 'Current configuration : 136 bytes\n'
 '!\n'
 'interface GigabitEthernet1\n'
 ' ip address 10.10.20.21 255.255.255.0\n'
 ' ip nat outside\n'
 ' negotiation auto\n'
 ' no mop enabled\n'
 ' no mop sysid\n'
 'end\n')
```

https://github.com/ktbyers/netmiko

netprog_basics/programming_fundamentals/python_part_3/api_netmiko_example.py

Data Types

Strings - a series of characters, surrounded by single or double quotes.

- “And now for something completely different…”
- "hello"+"world" "helloworld" # concatenation
- "hello"*3 "hellohellohello" # repetition
- "hello"[0] "h" # indexing
- "hello"[-1] "o" # (from end)
- "hello"[1:4] "ell" # slicing
- len("hello") 5 # size
- "hello" < "jello" True # comparison
- "e" in "hello" True # search

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types

String Functions

`str.lower(`***`string`***`)` - lowercase version of a string

`str.upper(`***`string`***`)` - uppercase version of a string

`str.isalpha(`***`string`***`)` - True if the string has only alpha chars

Many others: split, replace, find, format, etc.

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Printing with strings

Format method

Default_dinner = ”chicken"

print ("I’m in the mood for {} tonight.".format(Default_dinner))

>>>I’m in the mood for chicken tonight.

F-strings (version 3.6+)

Default_dinner = ”chicken"

print (f"I’m in the mood for {Default_dinner} tonight.")

>>>I’m in the mood for chicken tonight.

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types

Numbers

- Integer
  - 100
  - 1000
  - 10000
  - 0xff

- Float
  - 15.5
  - 1234.567

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types

Numbers

- Arithmetic operators we will use:

  `+ - * /` addition, subtraction/negation, multiplication, division

  `%` modulus, a.k.a. remainder

  `**` exponentiation

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

    `1 + 3 * 4` is `13`

  - Parentheses can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types

Numbers

- When integers and reals are mixed, the result is a real number.
  - Example: `1 / 2.0 = 0.5`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1***`, `***value2***`)`	larger of two values
`min(`***value1***`, `***value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

Math Functions

Use this at the top of your program:

```
from math import *
```

© 2013 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types (cont.)

Lists (think square) – stores items in a particular order

- **Sequenced:** Ordered sequential list of items
- **Heterogeneous:** Items in a list can be of different types
- **Mutable:** Items can be added, removed, inserted, popped, and sorted
- **Items:** List items can be data structures, to create lists of lists, lists of dictionaries or lists of tuples

- `MyList = [123, 'Hi', ['apple', 'banana', 'orange'], conf_file]`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Data Types (cont.)

Dictionaries (think curly)

- **Unordered:** Not sequenced, accessed by key (fast lookup)
- **Keys:** Key must be immutable & hashable—string, numeric, or tuple
  - Key:Value pairs
- **Mutable:** Items can be added, removed, or changed
- **Items:** Items can be simple or complex

(doesn't this look like JSON??)

-
```
          K       V
device1 = {"ip":"10.3.21.5",
"version":"A.01.01","username":"user1","password":"pw1”}
    K       V       K       V       K       V
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Booleans

A boolean value is either True or False. Variables with boolean values are often used to keep track of certain conditions within a program

Relational Operators

Operator	Meaning	Example	Result
==	equals	`1 + 1 == 2`	`True`
!=	does not equal	`3.2 != 2.5`	`True`
<	less than	`10 < 5`	`False`
>	greater than	`10 > 5`	`True`
<=	less than or equal to	`126 <= 100`	`False`
>=	greater than or equal to	`5.0 >= 5.0`	`True`

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Booleans

A boolean value is either True or False. Variables with boolean values are often used to keep track of certain conditions within a program

Logical Operators

Operator	Example	Result
and	`9 != 6 and 2 < 3`	True
or	`2 == 3 or -1 < 5`	True
not	`not 7 > 0`	False

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

If Statements

Used to test for particular conditions and respond appropriately

Can be nested

Watch out for indentions/white space – this affects scope

Syntax:

```
if <condition>:
    <statements>
```

```
x = 5
if x > 4:
        print(“x is greater than 4”)
print(“This is not in the scope of the if”)
```

cisco

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

If Statements (cont.)

```
if expression:
 statement(s)
elif expression:
 statement(s)
else:
  catchall
```

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Loops

For Loops are used to repeat a specific block of code a known number of times.

While loops run as long as certain conditions remaintrue

```
for statement
    for [variable] in [sequence]:
        do something
```

```
while statement
    while True:
        do something
```

cisco

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Functions

Functions are named blocks of code designed to do one specific job. They allow you to write code once that can then be run whenever you need to accomplish the same task. Functions can take in the information they need, and return the information they generate.

Example:

```
def my_function(x):
    do something
    return something
```

← This is optional

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Python TDD with pytest

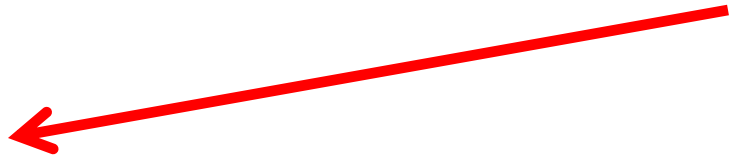
```
# content of test_step.py
import pytest

class TestUserHandling:
        def test_login(self):
                pass

        def test_modification(self):
                a = 2
                b = 3 - 1
                assert (a == b)

        def test_deletion(self):
                pass

        def test_normal():
                pass
```



© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

DevNet Code Exchange

Curated software built around Cisco platforms & APIs

Sample code

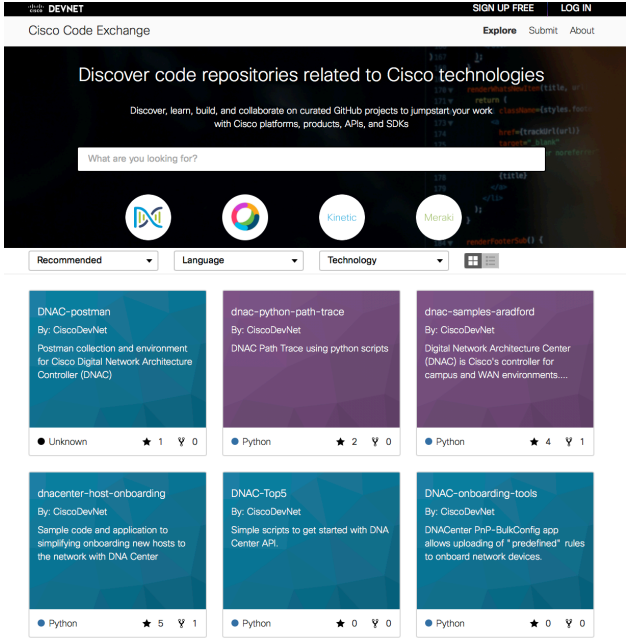
Connectors

Open source

Built on GitHub

Software written by the community

Connect to expert developers



https://developer.cisco.com/codeexchange

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

© 2019 Cisco and/or its affiliates. All rights reserved. Cisco Public

Python Challenge

Brad Ferguson

Bryce Breedlove

Buford Peek

Nick Grants

Bradley Knopps

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential


Play
Kahoot!

Questions?

© 2014 Cisco and/or its affiliates. All rights reserved. Cisco Confidential