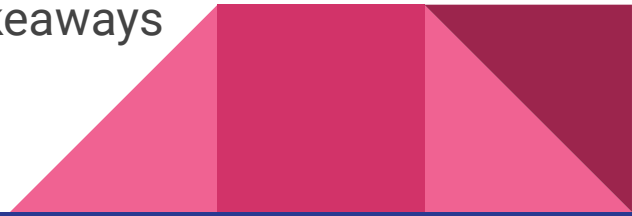
Python Beginner’s Workshop

In Collaboration with the Pikes Peak Library District 21st Century Library

Ryan Freckleton
PySprings: https://www.meetup.com/pysprings/
2019-JUN-06

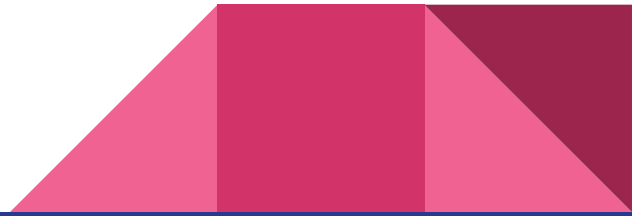
Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



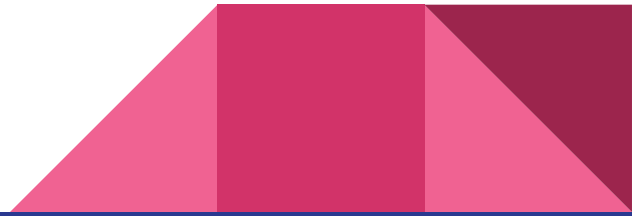
Context

- This is a safe environment to make mistakes and ask for help
- Most people wait too long before asking for help
- It’s good to make mistakes, we’re optimizing for learning, not aiming at perfection.
- Thank you for helping make this a welcoming, friendly event for all.
- Contact the organizers at pysprings@pysprings.org or https://sayat.me/pysprings (anonymous)



Greetings

1. Your name
2. How did you get here?



Learning Goals

1-2-4-All

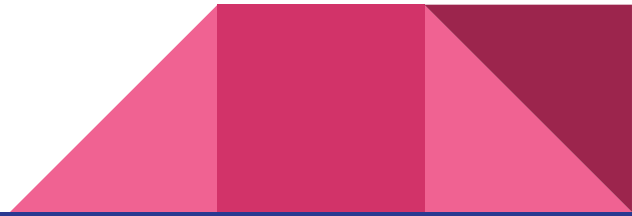
- What's one thing you know about programming in Python?
- What's one thing that you'd like to learn about programming in Python?

Learning Cycle

Introduction	Short lecture introducing a new concept from Python
Exploration	Hands-on application of the concept introduced. Work in groups and collaborate if you prefer! Explore the material in a hands-on manner
Invention	What have we learned through our exploration? What surprises did we encounter? What mysteries did we uncovered?
Application	With our newly “invented” knowledge, what can we do? This leads into a new exploration phase

What is Programming?

- Programming is a creative activity.
- It doesn’t involve much math (unless you want it to!)
- Programming is simply the act of entering instructions for the computer to perform.

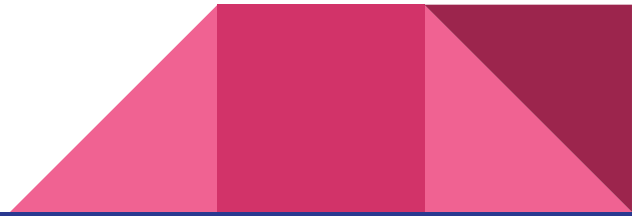


An Example

```
1  passwordFile = open('SecretPasswordFile.txt')
2  secretPassword = passwordFile.read()
3  print('Enter your password.')
4  typedPassword = input()
5  if typedPassword == secretPassword:
6     print('Access granted')
7     if typedPassword == '12345':
8        print('That one is used on luggage.')
9  else:
10    print('Access denied')
```

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Running Python from the command-line

Run Python's interactive prompt with:

```
$ python
```

Enter the following:

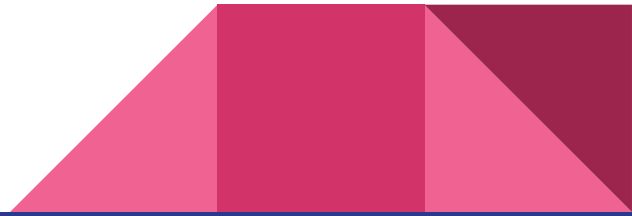
```
>>> print("Hello, World!")
```

Followed by:

```
>>> import this
```

Exit the interactive prompt with:

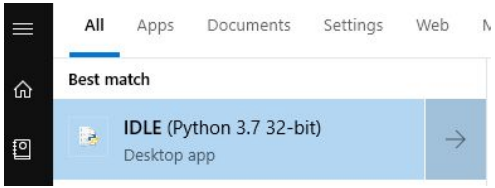
```
>>> exit()
```



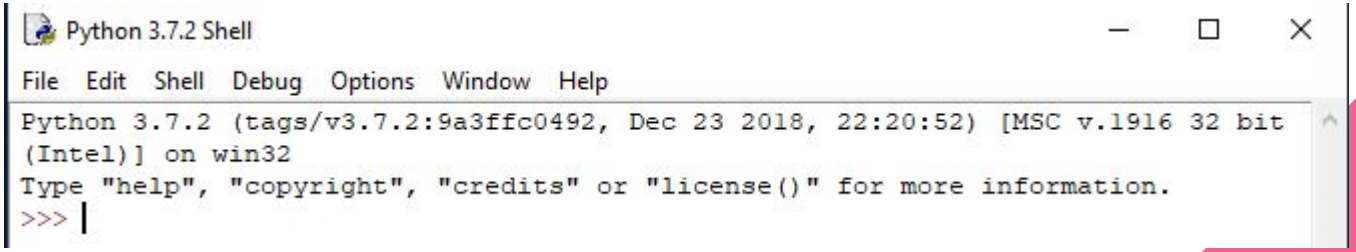
Running Python's IDLE Shell

PPLD computers come equipped with IDLE (python.org's IDE)

Press the "Windows" button and type in "idle". The application looks like this:



When you click the application, you'll be presented with this window:



Running Python’s IDLE Shell (cont)

In the window labeled “Python 3.x.x Shell” is what we call the “interactive prompt”. This window allows you to execute Python statements.

Try typing in the following statements. Remember: enter in everything *except* the “prompt” (>>>)

```
>>> print(“Hello, World!”)
>>> import this
```

Running a Python Script from the command-line

Let’s create a file named “script.py” and give it the following text:

```
print(“Hello, World!”)
```

Now open up “powershell.exe”, change to the directory you saved your script, and run it with:

```
$ python script.py
```

Running a Python Script from IDLE

From the IDLE shell, click on “File -> New File”. This will open up a text editor where you can enter in Python statements.

Enter the following statement into the editor:

```
print(“Hello, World!”)
```

Save the file to your Desktop by clicking “File -> Save”, clicking on “Desktop” on the left side of the pop up window, and entering “hello” in the “File name” section, then click on “Save”.

Now run your script by clicking on “Run -> Run Module”.

Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Notation

When you see an example like:

```
>>> print(“Hello, World!”)
```

It means “type it out in the interactive prompt.” Always ignore the “>>>” characters!

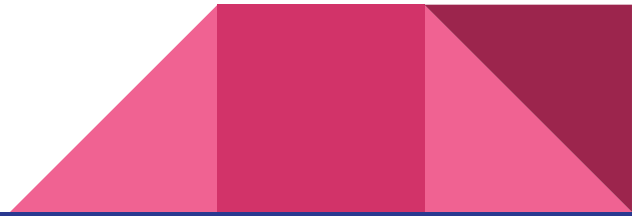
When you see an example like:

```
print(“Hello, World!”)
```

It means “type it out in a file and run it as a script.”

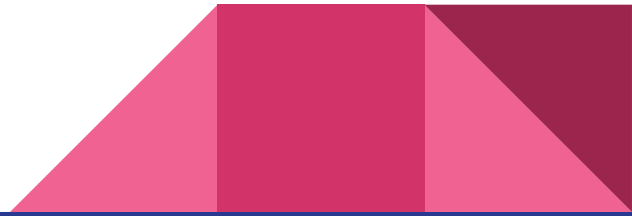
Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Python as a Calculator

```
>>> 100 * 2
200
>>> (1 + 2 + 3 + 4 + 5 + 6) / 6
3.5
>>> 1 - 2*100 + 3*12
-163
>>> abs(-163)
163
```



Python Math Operations

Operators:

- `+` `-` `*` `/`
- `%` `**` `//`
- Does python obey the order of operations?

Functions:

- **`abs bin hex oct ord round`**
- **`divmod min max pow`**
- What's the difference between these two lists of functions?

Python Math Operations

Operators:

- `+` `-` `*` `/`
- `%` `**` `//`
- Does python obey the order of operations?

Functions:

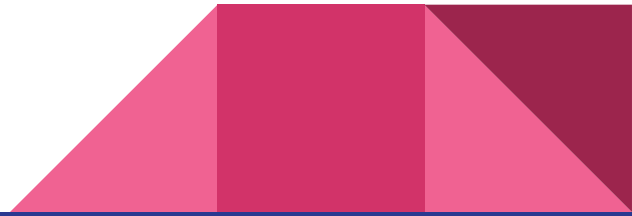
- **`abs bin hex oct ord round`**
- **`divmod min max pow`**
- What's the difference between these two lists of functions?

Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Functions

A function is a set of “reusable” instructions.

```
1  def hello():
2      print('Howdy!')
3      print('Howdy!!!')
4      print('Hello there.')
5
6  hello()
7  hello()
8  hello()
```

Functions

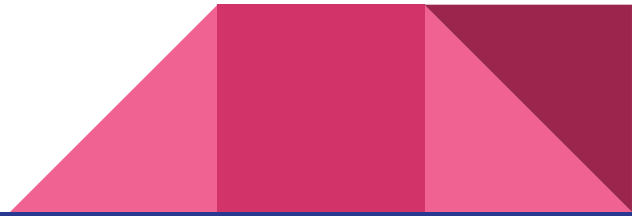
Function can take inputs too! We call these “parameters”.

```
1  def hello(name):
2      print('Hello', name)
3
4  hello('Alice')
5  hello('Bob')
```

Functions

Function can take any number of parameters and even return a result. Here's a function that takes 2 parameters and returns the result of adding them together:

```
1  def add(a, b):
2      return a + b
3
4  print(add(1, 2))
5  print(add(1, 2) + add(3, 4))
```

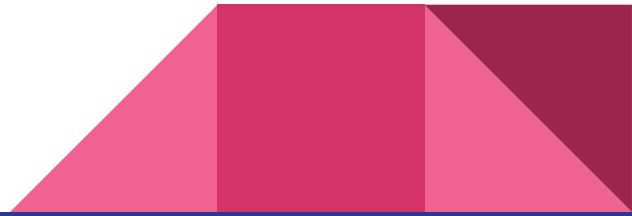


Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



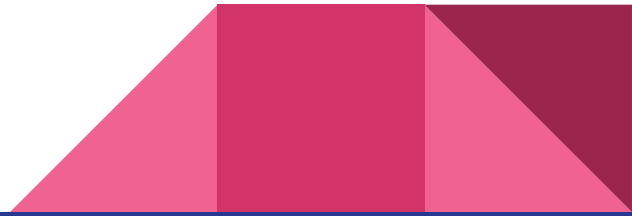
Strings

Examples

```
“This is a string.”
‘This is also a string.’
“This is ‘a’ string.”
‘This is “a” string.’
“This is an \”ugly\” string!”
```

We can also get more information from python:

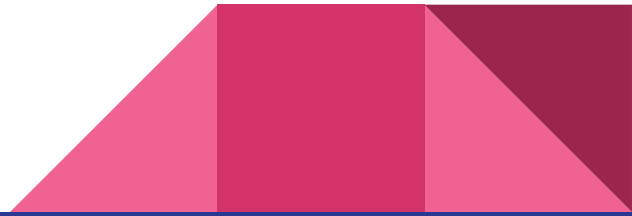
```
>>> help(str)
```



Strings

More Examples

```
>>> 'this is a string'.title()
'This Is A String'
>>> 'this is a string'.upper()
'THIS IS A STRING'
>>> 'what ARE you doing!?'.lower()
'what are you doing!?'
>>> "   there's whitespace in this    ".strip()
"There's whitespace in this string."
```



Hello again

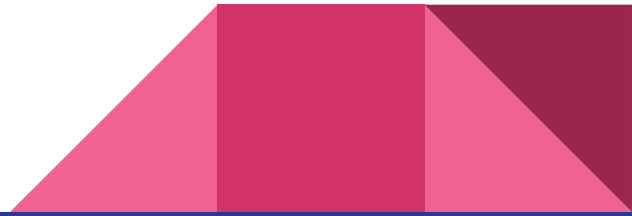
```
name = input('What is your name? ')
print('Hello, ' + name + '!')
```

Let's try it!

If you have command line access:

```
$ python hello.py
```

Or use IDLE's "Run -> Run Module (F5)" command



Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

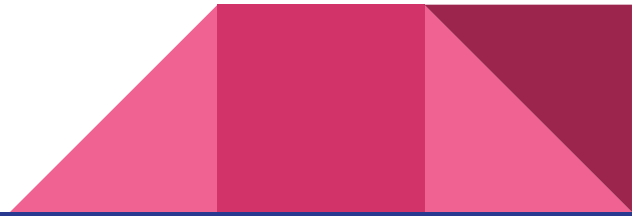
String Indexing and Slicing

```
>>> s = 'We are the Knights who say ni!'
>>> s[0]
'W'
>>> s[-1]
'!'
>>> s[7:10]
'the'
>>> s[-7:-4]
'say'
```

String Indexing

```
 +---+---+---+---+---+---+
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

```
>>> s = 'Python'
>>> s[1:4]
'yth'
>>> s[5]
'n'
```



Strings

What are the methods of list?

Show the full help text for strings:

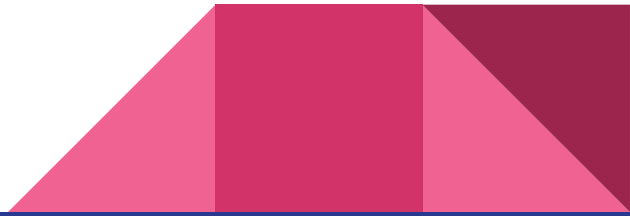
```
>>> help(str)
```

Show the help text for a single function:

```
>>> help(str.lower)
```

Also try out:

```
>>> dir(str)
```

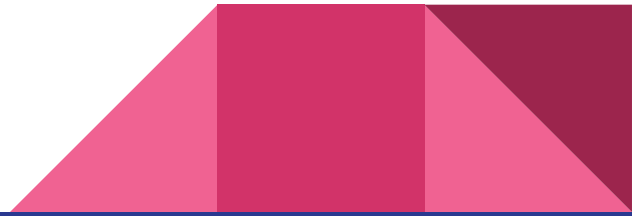


Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



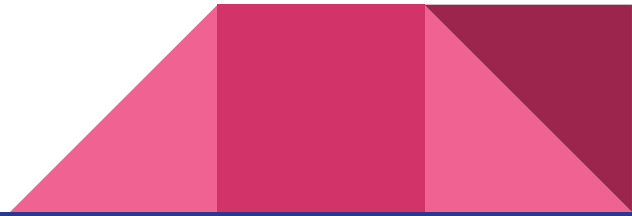
Numbers

There are two basic types of numbers in Python:

`int` This is the “integer” type. Think of these as whole numbers: 1, 42, 10000000

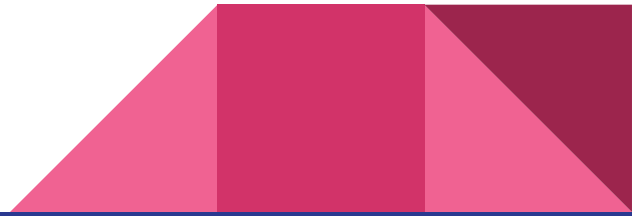
`float` This is the “floating point” type. These are *non-whole* numbers: 3.1415, 9.99

```
>>> -1 / 4
-0.25
>>> 1 // 4
0
>>> 1 + 4
5
>>> 1 + 4.5
5.5
```



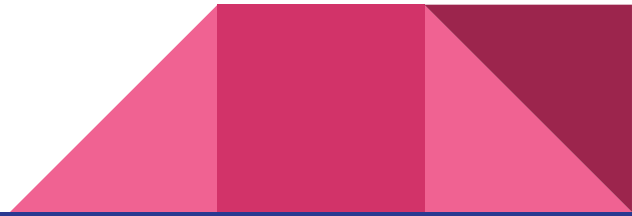
Outline

Introduction
First Steps
    Running Python
    Expressions
    Functions
Data Types
    Strings
    Numbers
    Lists
    Dictionaries
Libraries
    Environments
    Third-Party Packages
Control Flow
    Booleans
    Looping and Branching
Conclusion
    Practice Problems
    Final Takeaways
    Projects!



Lists

```
>>> mylist = [1, 2, 'three', "4", 5.3]
>>> s = "What are the words in this string?"
>>> s.split()
['What', 'are', 'the', 'words', 'in', 'this', 'string?']
>>> words = s.split()
>>> words.sort()
>>> words
['What', 'are', 'in', 'string?', 'the', 'this', 'words']
```



Lists

What are the methods of list?

Remember:

```
>>> help(list)
```

Also try out:

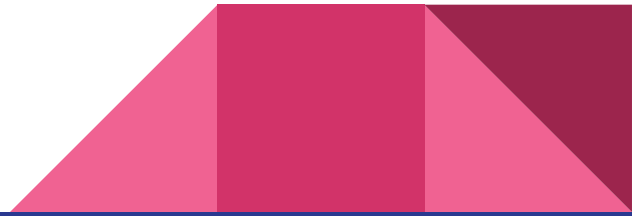
```
>>> dir(list)
```

Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



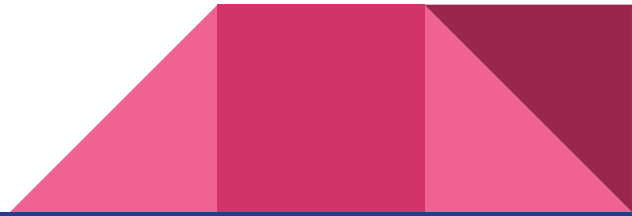
Dictionaries

Dictionaries are like lists, but they have “key value” pairs.

```
>>> myCat = {'size': 'tiny', 'color': 'gray', 'disposition': 'loud'}
>>> myCat['size']
'tiny'
>>> "My " + myCat['size'] + " cat has " + myCat['color'] + " fur."
'My tiny cat has gray fur.'
```

Dictionaries have methods, just like “str.lower()”

```
>>> myCat.keys()
dict_keys(['size', 'color', 'disposition'])
>>> myCat.values()
dict_values(['tiny', 'gray', 'loud'])
>>> myCat.items()
dict_items([('size', 'tiny'), ('color', 'gray'),
('disposition', 'loud')])
```



Dictionaries

What are the methods of dict?

Remember:

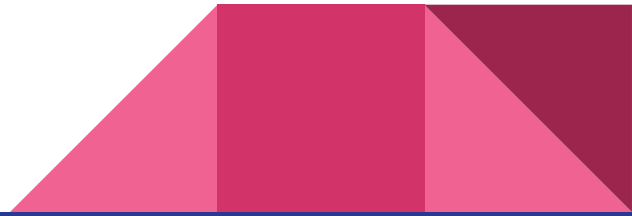
```
>>> help(dict)
>>> dir(dict)
```

Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Environments

Python uses *environments* to keep projects separate.

NOTE: This does not (currently) work on PPLD computers!

One way to do this is to use “virtualenv”. Create the environment with:

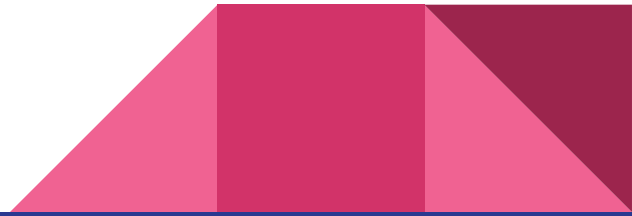
```
$ virtualenv raindrop
```

Windows:

```
$ raindrop\Scripts\activate
```

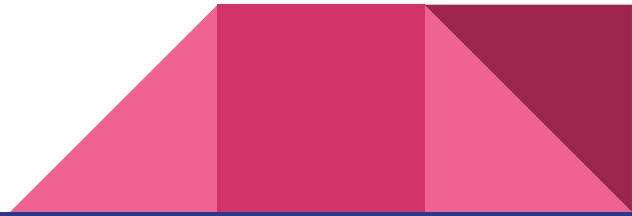
Linux and OSX:

```
$ . raindrop/source/bin/activate
```



Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Installing Third-Party Packages

We generally use the “pip” command-line application to install third-party packages.

```
$ pip install requests
```

This should install the requests library into your environment.

NOTE: This does not (currently) work on PPLD computers!

Requests Example

This script will talk to a website which returns your public IP address.

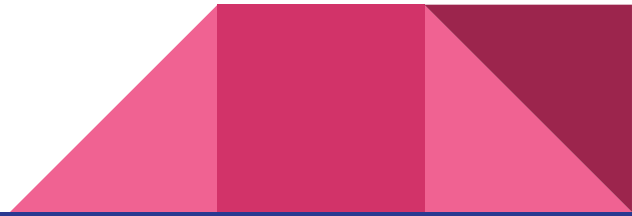
```
import requests
resp = requests.get('http://httpbin.org/ip')
print(resp.json())
```

NOTE: This does not (currently) work on PPLD computers!

Finding Third-Party Packages

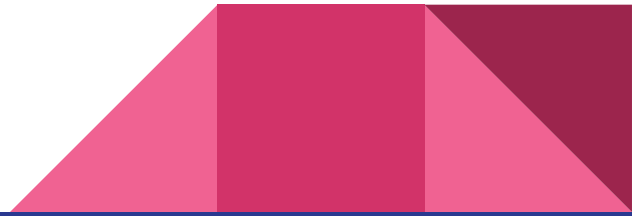
Here are the websites that house *most* Python packages

- https://pypi.org (newer)
- https://pypi.python.org (original)



Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Booleans

```
>>> bool(1)
True
>>> bool(0)
False
>>> bool("no")
True
>>> bool("")
False
>>> bool([])
False
>>> bool([42])
True
```

Booleans

You can combine boolean operations with “and” and “or”.

```
>>> True or False
True
>>> True and False
False
>>> True and False or True and True or False
True
>>>> (True or False) and (False or True)
True
>>> (True or False) and (False and True)
False
```

Booleans

There are a few more “boolean operators” that we can use.

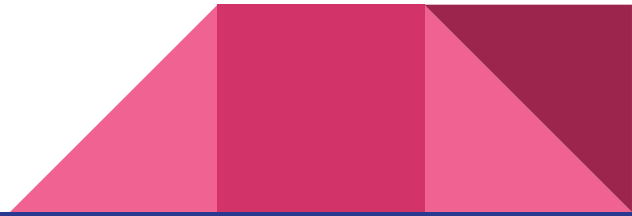
`==`	“Is equal to”.	Example: `2 == 2`
`!=`	“Is NOT equal to”.	Example: `2 != 3`
`<=`	“Is less than or equal to”.	Example: `2 <= 3`
`>=`	“Is greater then or equal to”.	Example: `3 >= 2`
`>`	“Is greater than”.	Example: `10 > 1`
`<`	“Is less than”.	Example: `1 < 10`
`in`	“Is the needle in the haystack”.	Example: `'i' in 'Tim'`

Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Looping and Branching

We use the “if/else” conditional to test some value.

```
password = input(“Enter the secret word: “)
if password == “sesame”:
    print(“Access granted.”)
else:
    print(“Access denied!”)
```

Looping and Branching

We can extend our “if/else” to any number of conditions using one or more “elif” clauses.

```
age = int(input("How old are you? "))
if age < 18:
    print("You're not old enough to dance.")
elif age == 18:
    print("Welcome, is this your first time here?")
else:
    print("You can dance if you want to, you can leave "
          "your friends behind.")
```

Looping and Branching

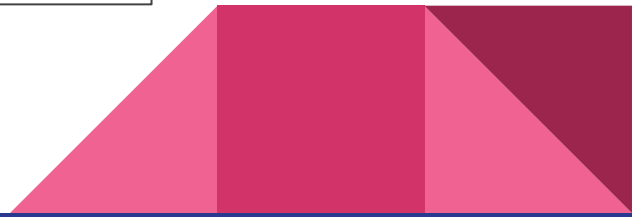
The “for” loop is used when you want to loop over a collection of things.

```
>>> words = 'this is a list of words'.split()
>>> for word in words:
...     print(word.title())
...
This
Is
A
List
Of
Words
```

Looping and Branching

The “while” loop is used when you aren’t sure when to stop.

```
while True:
    password = input(“Enter the secret word: “)
    if password == “sesame”:
        print(“Access granted.”)
        break
    else:
        print(“Access denied!”)
```

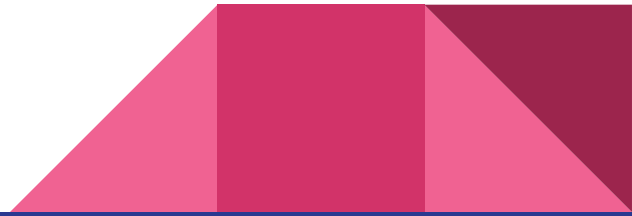


Invention

- What problems, if any, did you encounter?
- What mysteries, if any, did you encounter?
- What other takeaways are there from this session?
- What could you use from it in the future?

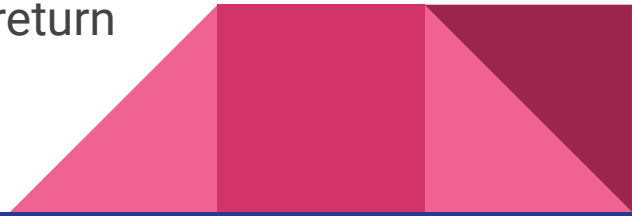
Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Practice Problems

- Write code that prints **Hello** if 1 is stored in the variable *spam*, prints **Howdy** if 2 is stored in the variable *spam*, and prints **Greetings!** if anything else is stored in the variable *spam*.
- Write a short program that prints the numbers 1 to 10 using a **for** loop. Then write an equivalent program that prints the numbers 1 to 10 using a **while** loop.
- Write a *function* named **collatz()** that has one parameter named **number**. If the value of **number** is even, then **collatz()** should print and return **number//2**. If the number is odd, then **collatz()** should print and return **3*number+1**



Practice Problems

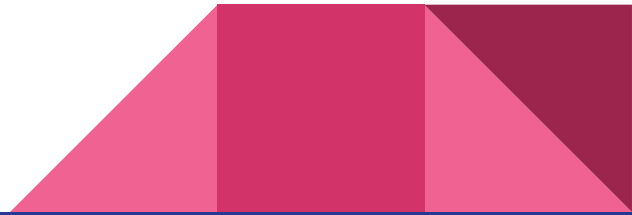
Say you have defined the following list:

```
spam = ['apples', 'bananas', 'tofu', 'cats']
```

Write a **function** that takes a list value as an argument and returns a string with all the items separated by a comma and a space, with the work “and” inserted before the last item.

For example, passing **spam** as defined above, the function would return the string **“apples, bananas, tofu, and cats”**. But your function should be able to work with any list value passed to it!

HINT: `help(str.join)`



Practice Problems

You are creating a fantasy video game. The data structure to model the player's inventory will be a **dictionary** where the keys are string values describing the item in the inventory and the value is an integer value detailing how many of that item the player has. For example, the dictionary value

```
{'rope': 1, 'torch': 6, 'gold coin': 42, 'dagger': 1,
 'arrow': 12}
```

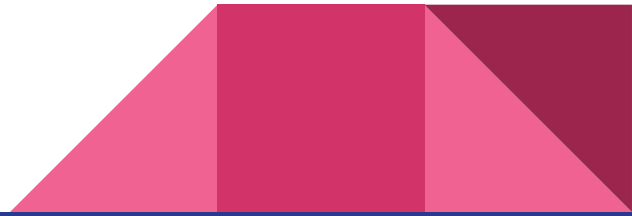
means the player has 1 rope, 6 torches, 42 gold coins, and so on.

(continued on next slide)

Practice Problems

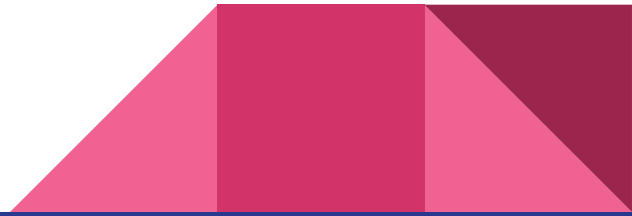
Write a function named **displayInventory()** that would take any possible “inventory” dictionary and display it like the following:

```
Inventory:
12 arrow
42 gold coin
1 rope
6 torch
1 dagger
Total number of items: 62
```



Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!

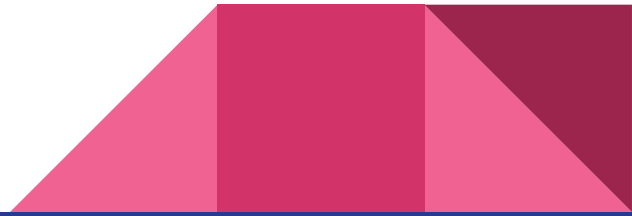


Conclusion

- Final Takeaways (1-2-4-all)
- Survey: https://goo.gl/forms/ZpNl0z8pw5J8J8Rv1
- Anonymous feedback: https://sayat.me/pysprings
- Material based on https://automatetheboringstuff.com/ released under CC BY-NC-SA 3.0

Outline

- Introduction
- First Steps
  - Running Python
  - Expressions
  - Functions
- Data Types
  - Strings
  - Numbers
  - Lists
  - Dictionaries
- Libraries
  - Environments
  - Third-Party Packages
- Control Flow
  - Booleans
  - Looping and Branching
- Conclusion
  - Practice Problems
  - Final Takeaways
  - Projects!



Projects!

Here are some resources we thought you could use to practice your Python.

- Reddit Daily Programmer: https://www.reddit.com/r/dailyprogrammer/
  - Game of Threes https://redd.it/3r7wxz
  - Rövarspråket (Robber's Language) https://redd.it/341c03
- WordPlay: https://github.com/jesstess/Wordplay
- Colorwall: https://github.com/jesstess/ColorWall