* Course Overview
* Variable and Numbers
  + Data science, machine learning, web development and more
  + There’s a package for everything
  + Interpreted language
  + Python interpreter let’s you run python lines one at a time
  + Primitive data type
    - Integers: int
  + Python infers the type
* Demo: Install Python
  + In command line
    - Type ‘python --version’
  + Install python
    - Go to python.org and download
  + Install visual studio code
  + Install python extension, Python for VSCode in visual studio code
  + Then tell vs code how to run python program by telling it which interpreter to use
    - View -> command palette
    - Search python select interpreter
    - Use python 3.9
* Demo: Create a Tax Calculator
  + Scripts saves code
  + Files ending in .py is called a python script or python program
  + print(): output to screen
* Strings, Input, and Output
  + data type conversion
    - make a call to the int() function
    - ex) amount = int(10.6)
  + a String stores text
    - can use ‘’ or “”
    - double quotes can be useful if a single quote is literally part of the String
  + string concatenation using + between two strings
  + use input() function to get input from user
    - will store as string
    - ex) my\_name = input(“What’s your name?”)
    - the message gets printed to the screen
    - the program waits for the user to input something and press enter
  + \n is a special character for a newline
  + int are whole numbers
  + floats which are decimals
  + string stores text
* Demo\_ Crate an Age Calculator
  + //: for integer division
  + % for remainder
* Conditionals
  + a conditional statement, or if statement, let us make decisions in python
  + 6 python comparators: <, <=, ==, >=, >, !=
  + whitespace indents in python need to be consistent, otherwise there will be an IndentationError
  + if, elif, else
    - if(condition): ..
    - elif (condition):…
    - else:
  + or: lets you combine multiple comparisons
    - if temperature > 80 or temperature < 60:
  + and: both comparisons need to be True for the if statement to be True
    - let you combine multiple comparisons
    - if temperature < 80 and forecast != “rain”:
  + not: lets you negate a comparison
    - if not forecast == “rain”:
  + 3 python logical operators: or, and, not
  + a boolean can store True or False value
  + all primitive data types: int, float, string, boolean
* Demo: Rock, Paper, Scissors G…
* Import: Python Modules
  + When you install python
    - You get the python interpreter
    - You get built in types and functions
    - You also get the python standard library
  + If you need something extra you can import it from the python standard library
    - Ex) import random
  + Need to import the module to use it
* Lists and Loops
  + A list is a container of things
  + Can store anything you want
  + Can have mixed items
  + An item’s index is its position
    - Index starts a 0
  + Use List.append() to add items to list
    - acronyms = []
    - acronyms.append(‘LOL’)
  + can also create a list with initial items
    - acronyms = [‘LOL’, ‘IDK’, ‘SMH’]
  + List.remove() to remove items from the list
    - acronyms.remove(‘LOL’)
  + use can use either remove or del depending on whether you know the value or the index
    - use remove if you know the value
    - use del if you know the index
  + use if .. in .. to check if an item exists in list
    - if 1 in [1, 2, 3, 4, 5]:
    - print(‘True’)
  + For loop
    - for acronym in acronyms:
    - print(acronym)
* Demo: Sum Expenses
  + print()
    - can separate value with commas, they will be separated by “ “ in the output
    - ex) print(‘You spent’, sum)
    - you can also specify what separate the values by setting sep
    - ex) print(‘You spent $’, sum, sep=’’)
  + sum()
    - sums the value of the list
* Loops with range()
  + range()
    - generates and returns a sequence
    - exclusive
    - range(7) generates sequence (0, 1, 2, 3, 4, 5, 6)
    - range(start, end, step)
* Demo: Loan Payment Calculator
  + #: comments in python
  + print()
    - set value of end to determine what is outputted at the end
    - print(“test”, end=’ ‘)
* Dictionaries
  + A dictionary maps keys to values
    - acronyms = { ‘LOL’: ‘laugh out loud’, ‘IDK’: ‘I don’t know’ }
  + each item is know as a key-value pair
  + to look up a value in a dictionary we send in a key
    - acronyms[‘LOL’]
  + dictionaries can hold anything, can mix
  + create an empty dictionary
    - acronyms = {}
  + adding new dictionary items
    - acroynms[‘LOL’] = ‘laugh out loud’
  + order is random in a dictionary
  + updating a value is the same way a value is added
    - acronyms[‘LOL’] = ‘Laugh Out Loud’
  + removing a dictionary item
    - del acronyms[‘LOL’]
  + trying to access a key that doesn’t exist will cause an error, a KeyError
  + use get() to avoid KeyError
    - acronyms.get(‘BTW’)
    - if key is not present will get a None type instead of an error
  + None type
    - Represents the absence of a value
    - Evaluates to False in a conditional
* Demo: Create a Movie Schedule
  + Iterate over the keys in a dictionary
    - for key in current\_movies:
* Combining Lists and Dictionar…
  + Can use two indexes to get an individual item from a two-dimensional list
    - menus = [ [‘Egg Sandwich, ‘Bagel’], [‘BLT, PB&J’]]
    - menus[0][1]
  + dictionary of lists
    - menus = { ‘Breakfast’ : [‘Egg Sandwich’, ‘Bagel’],
    - ‘Lunch’: [‘BLT’, ‘PB&J’] }
  + For looping over dictionaries defaults to just returning the keys in a dictionary
  + Use items() to get key and value
    - for name, menu in menus.items():
    - print(name, ‘:’, menu)
  + Can use dictionaries to represent objects
* Demo: Parse a Nested Contac..
* Reading JSONS and Installing …
  + make http request and get http response
  + some websites return raw data
    - usually the raw data is returned under the API(Application Programming Interface) for the website such as api.twitter.com
  + JSON Data
    - data format used to exchange data to and from a web server
    - common use of JSON is to exchange data to/from a web server
  + JSON format can be a mix of lists and dictionaries
  + JSON(Javascript Object Notation)
    - json started in JavaScript but now can be used in any programming language
  + need the requests library to do an http request in python
    - need to install it ourselves
  + pip
    - used to install any package from the python package index
  + pip --version
    - return the version of pip installed
  + pip install request
    - to install the request library
  + open-notify.org
    - api returns the current number of people in space
    - if available it also returns the names and spacecraft those people are on
  + have to import the request module
  + ex)
    - import requests
    - response = requests.get(‘<http://api.open-notify.org/astros.json>’)
    - json = response.json()
* Demo: Create a Python Virtual…
  + can use virtual environments to handle different versions of packages
  + install virtualenv
    - py -m pip install --user virtualenv
  + create virtual environment
    - python -m venv [VM name]
  + to activate the VM
    - .\[vm name]\Scripts\activate
* Demo: Use the Open Weather
  + Get API key from OpenWeather website and url from the API page
  + Ex)
    - import requests
    - api\_key = ” dae5f6326757b1f052a57f542b1aa3b5”
    - city = “Sacramento”
    - url = “http://api.openweathermap.org/data/2.5/weather?q=”+city+”&appid=”+api\_key
    - request = requests.get(url)
    - json = request.json()
* Functions
  + Functions are like mini-programs that complete a specific task
  + Functions start with the def keyword
  + Ex)
    - def greeting(name):
    - print(‘Hello’, name)
  + Orders matter, the functions need to be defined first before you use them
  + Scope
    - A variable created inside a function can only be used inside that function
    - This is called local scope
    - A variable created in the main body of the program is global variable and has global scope
    - That means it can be used anywhere
  + Using global variables can become messy
  + Reasons to create a function
    - Reuse that chunk of code over and over
    - You want to organize your code by logical units
* Demo: Create a Dice Rolling G…