An Introduction to Data Analysis

Josh Bodyfelt, Ph.D.

- Oct. 29 – Python Essentials
- Oct. 30 – Theory of Databasing

An Introduction to Data Analysis

Josh Bodyfelt, Ph.D.

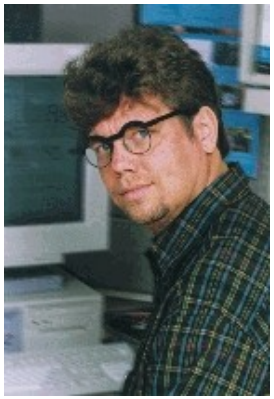
- Oct. 29 – Python Essentials
- Oct. 30 – Theory of Databasing
- Nov. 6 – CRUD'ing with SQLite
- Nov. 19 – Probability is Probably Useful

An Introduction to Data Analysis

Josh Bodyfelt, Ph.D.

- Oct. 29 – Python Essentials
- Oct. 30 – Theory of Databasing
- Nov. 6 – CRUD'ing with SQLite
- Nov. 19 – Probability is Probably Useful
- Nov. 26 – A Picture is Worth 1000 Words
- Nov. 27 – Ground Control to Major Tom

Python Essentials: **WHAT** is Python?

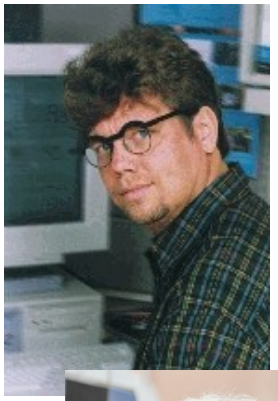


- 1989: Christmas – Guido van Rossum

  @ Centrum Wiskunde & Informatica, Amsterdam

- 1999: DARPA Grant
  - *An easy and intuitive language just as powerful as major competitors*
  - *Open source, so anyone can contribute to its development*
  - *Code that is as understandable as plain English*
  - *Suitability for everyday tasks, allowing for short development times*

Python Essentials: **WHAT** is Python?



- 1989: Christmas – Guido van Rossum

  @ Centrum Wiskunde & Informatica, Amsterdam

- 1999: DARPA Grant
  - *An easy and intuitive language just as powerful as major competitors*
  - *Open source, so anyone can contribute to its development*
  - *Code that is as understandable as plain English*
  - *Suitability for everyday tasks, allowing for short development times*

- 2008: Python 3.0 Released
- 2020: Python 3.9.0 Released Oct. 5th
- 2023: Python 4.0 Scheduled

Python Essentials: Executive Summary?

“Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.”

Python Essentials: **WHAT** is Python? **WHY** use it?

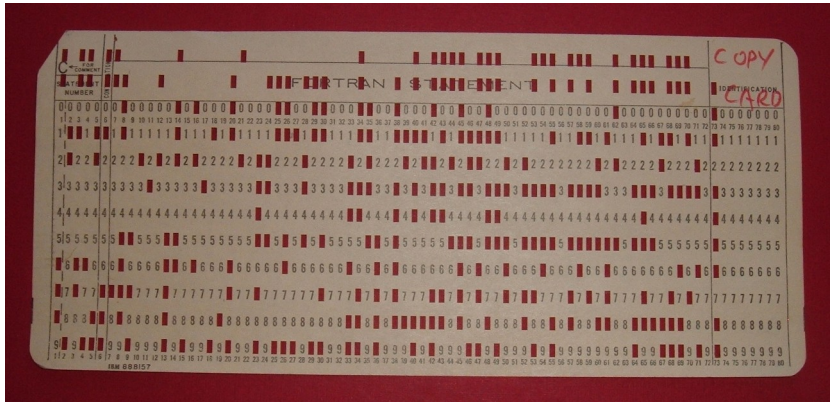
- High-Level & Interpreted
- Dynamic Semantics & Typing
- Built-In Data Structures
- Object-Oriented
- Modules & Packages

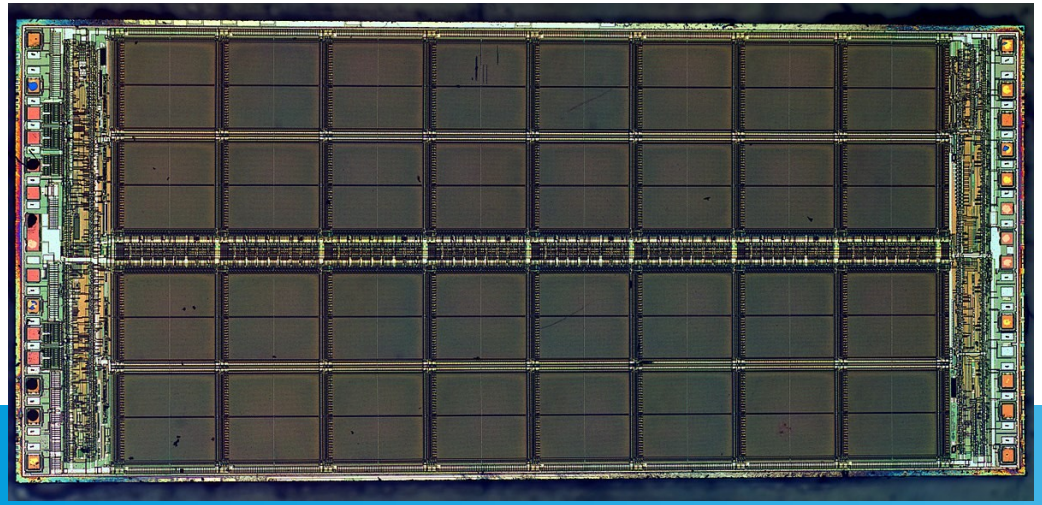
- Rapid App Development
- Scripting & Gluing
- Readability (“PEP8”)
- Open Source

Python Essentials: **WHAT** is Python?

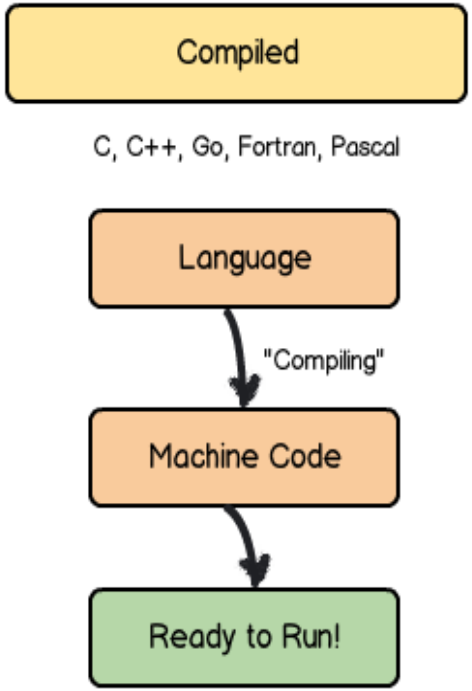
- High-Level
  - No “Bare Metal” Assembly or Binary
  - No Memory Mapping – Automatic!
  - Focus on **Abstract Objects**

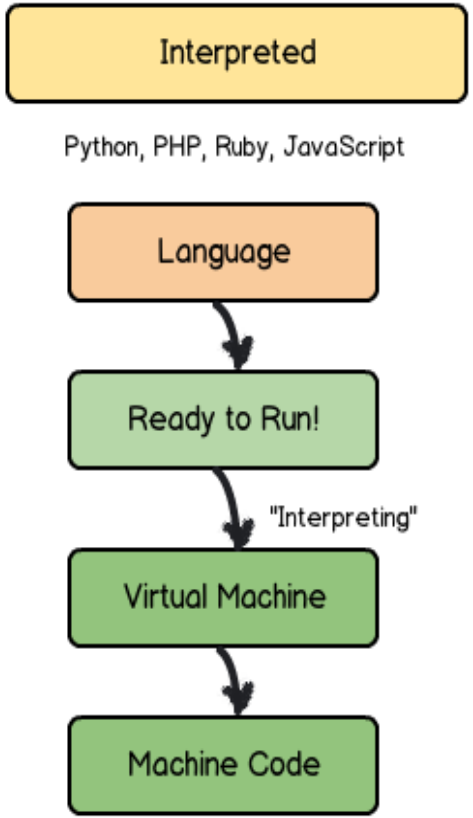
    ‘variable’, ‘array’, ‘expression’, ‘function’, ‘class’





Python Essentials: **WHAT** is Python?



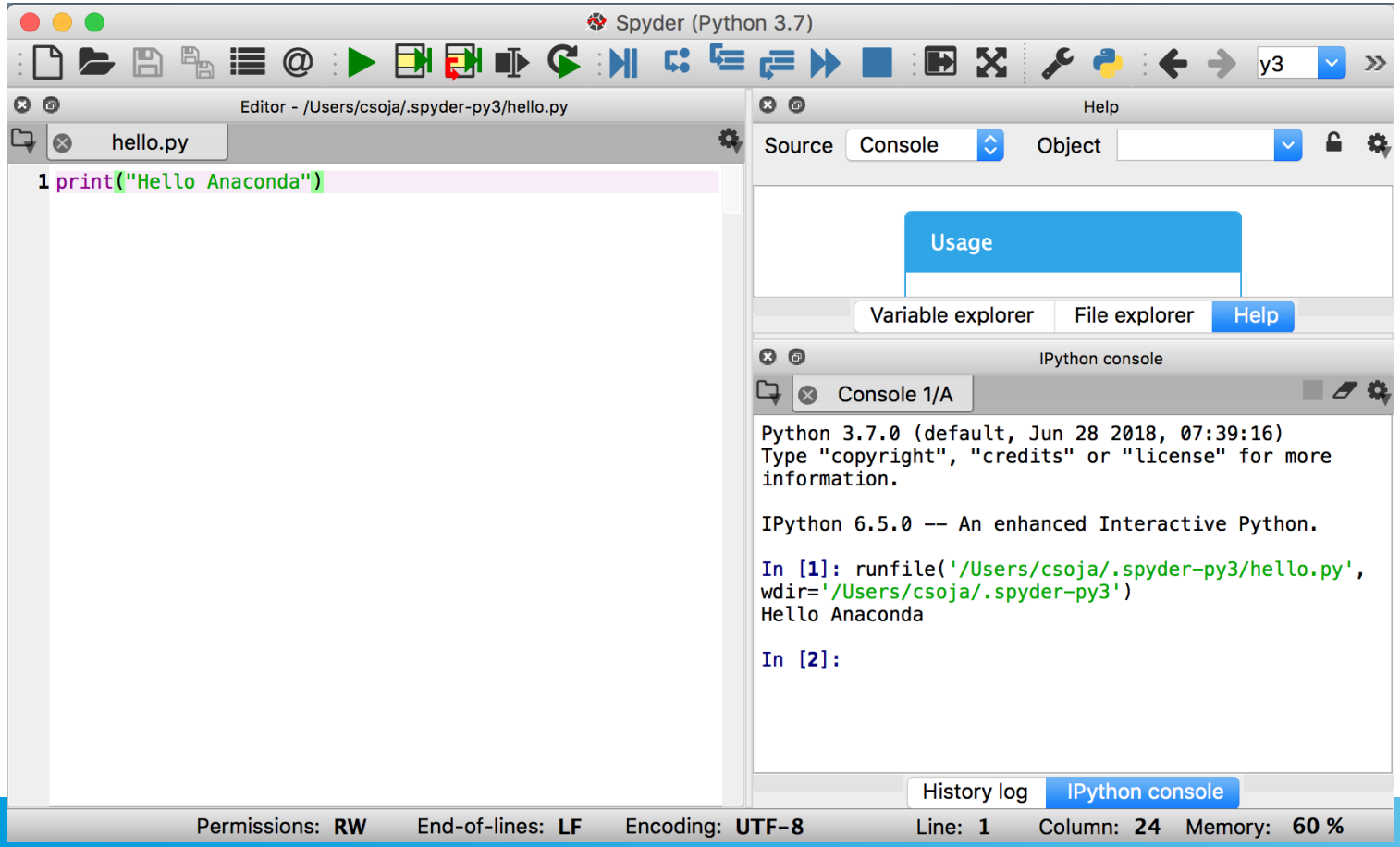


- Interpreted
  - Incredibly flexible
  - Runs on most OS
  - Allows for several IDEs

Python Essentials: Open Source

ANACONDA®

Python Essentials: Scripts vs. REPL (“Shell”)



- Read
- Evaluate
- Print
- Loop

Python Essentials: Basic Structures

- Dynamic Typing

```
> A=32
> type(A)
```

```
> B=True
> type(B)
```

```
> C=101.433
> type(C)
```

```
> D=”Favorite Things”
> type(D)
```

- Dynamic Casting

```
> E=A+C
> type(E)
```

```
> F=A+C
> type(F)
```

Python Essentials: Basic Structures

- Dynamic Typing

```
> A=32
> type(A)
```

```
> B=True
> type(B)
```

```
> C=101.433
> type(C)
```

```
> D=”Favorite Things”
> type(D)
```

- Dynamic Casting

```
> E=A+C
> type(E)
```

```
> F=A+C
> type(F)
```

```
> F=str(A)+C
> type(F)
```

Python Essentials: Basic Structures

- Built-In Data Structures

```
> fruit = ("apple", "banana", "cherry", "orange", “kiwi", "melon", "mango")
> print(fruit); type(fruit)
> print(fruit[0]); print(fruit[-1])
> fruit[-1]="tangerine"
```

Tuple

Python Essentials: Basic Structures

- Built-In Data Structures

```
> fruit = ("apple", "banana", "cherry", "orange", “kiwi", "melon", "mango")
> print(fruit); type(fruit)
> print(fruit[0]); print(fruit[-1])
> fruit[-1]="tangerine"
```

Tuple

```
> fruit = ["apple", "banana", "cherry", "orange", “kiwi", "melon", "mango"]
```

List

Python Essentials: Basic Structures

- Built-In Data Structures

```
> fruit = ("apple", "banana", "cherry", "orange", “kiwi", "melon", "mango")
> print(fruit); type(fruit)
> print(fruit[0]); print(fruit[-1])
> fruit[-1]="tangerine"
```

Tuple

```
> fruit = ["apple", "banana", "cherry", "orange", “kiwi", "melon", "mango"]
```

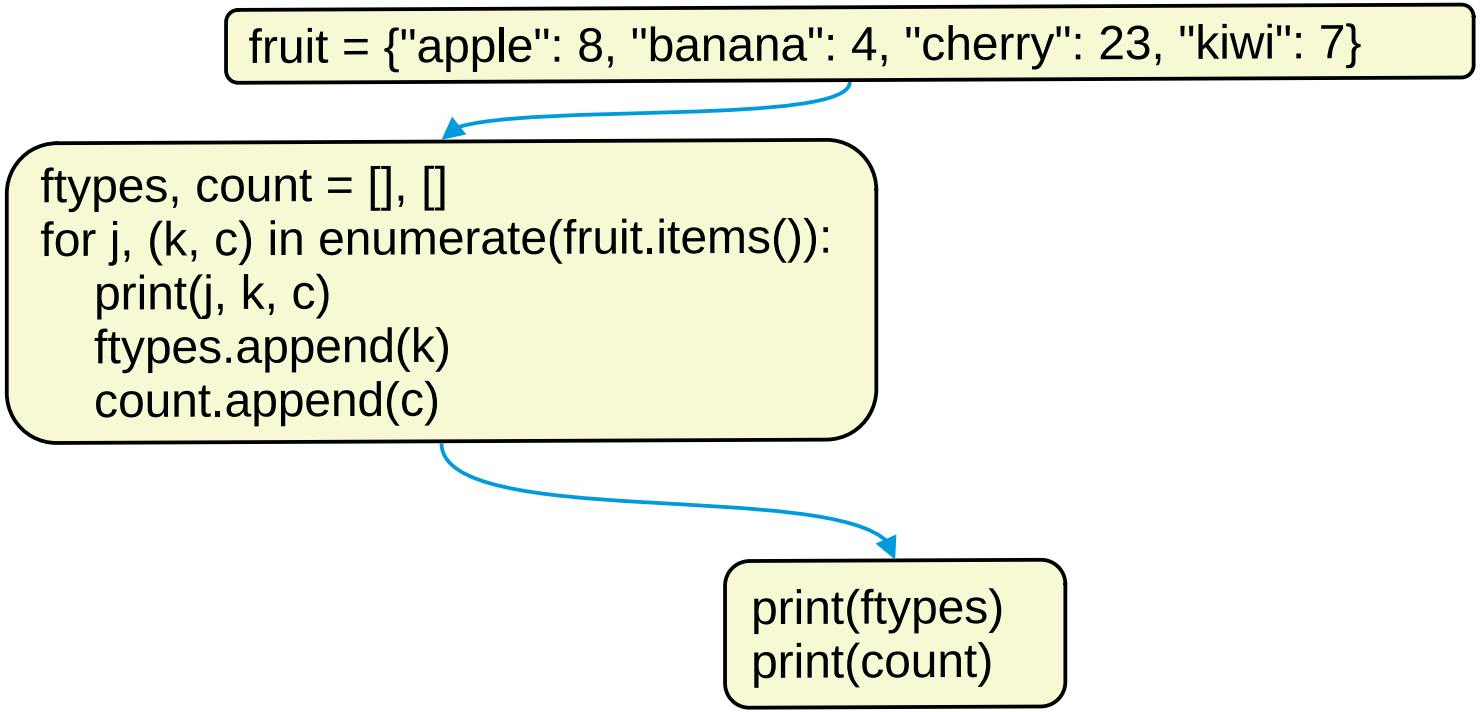
List

```
> fruit = {"apple": 8, "banana": 4, "cherry": 23, “kiwi": 7}
> fruit.keys()
> fruit.values()
> fruit.items()
> tuple(fruit); list(fruit.items())
```

Dictionary

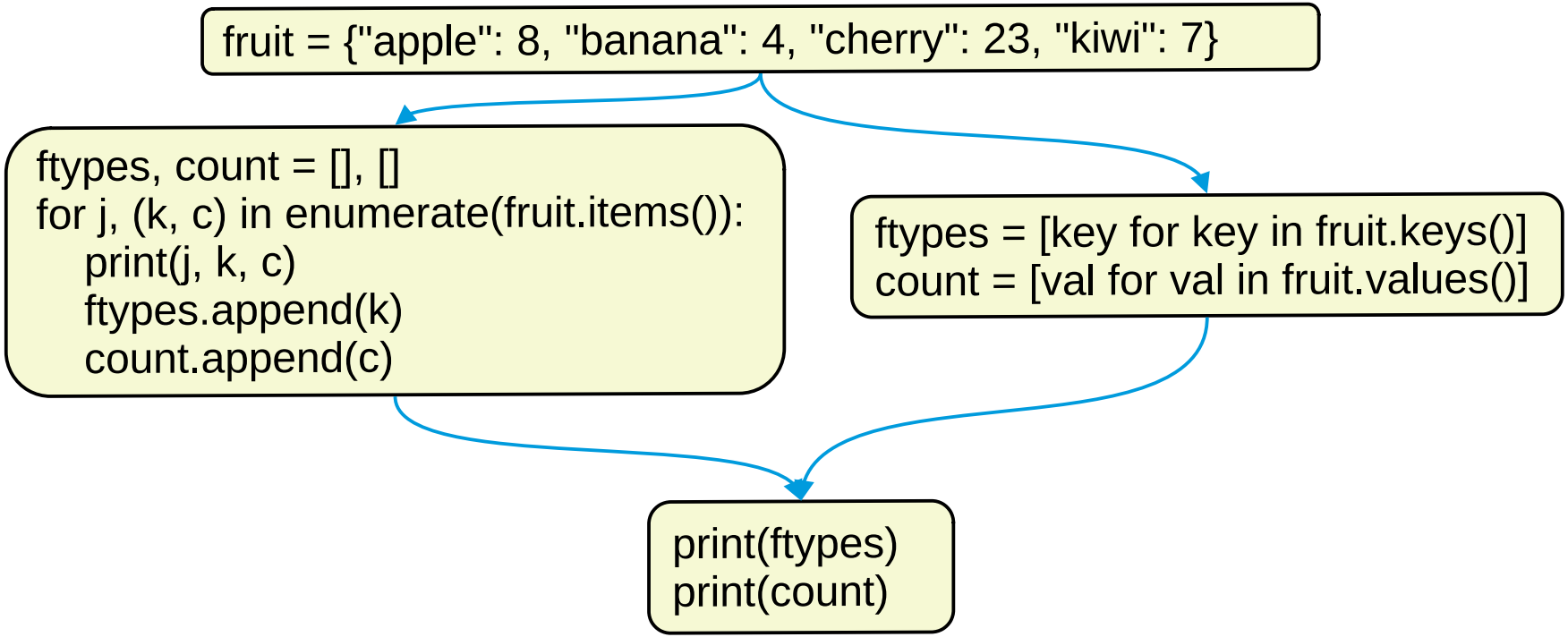
Python Essentials: Basic Structures

- Dynamic Semantics: Comprehension



Python Essentials: Basic Structures

- Dynamic Semantics: Comprehension



Python Essentials: Basic Structures

- Dynamic Semantics: Functions

```
def basket(names, counts):
    return {key: val for key, val in zip(names, counts)}

fruits=[“apple”, “banana”, “cherry”, “kiwi”]
counts = [8,4,23,7]
print(basket(fruit, counts))
```

Python Essentials: Basic Structures

- Dynamic Semantics: Functions

```
def basket(names, counts):
    return {key: val for key, val in zip(names, counts)}

fruits=[“apple”, “banana”, “cherry”, “kiwi”]
counts = [8,4,23,7]
print(basket(fruit, counts))
```

- Dynamic Semantics: Modules

```
def basket(names, counts):
    return {key: val for key, val in zip(names, counts)}

def total(counts):
    return sum(counts)
```

my_mod.py

```
> import my_mod as mm
> names = [“orange”, “lime”, “grape”]
> counts = [22, 4, 67]
> basket = mm.basket(names, counts)
> total = mm.total(counts)
> print(basket)
> print(total)
```

Python Essentials: Advanced Structures

- Object-Oriented

```
class shopCart:
    def __init__(self, customer_name):
        self.name = customer_name
        self.cart = {}
    def add_item(self, name, cost, quantity):
        if(name in self.cart.keys()):
            self.cart[name].append([cost, quantity])
        else:
            self.cart[name] = [[cost, quantity]]
    def print_cart(self):
        total = 0.0
        for name, prices in self.cart.items():
            subtot = 0.0
            print("\n"+name.upper()+"S:")
            print("\tUnits\tCost\tSubtotal")
            for (cost, quant) in prices:
                print(f"\t{quant:d}\t${cost:.2f}\t${quant*cost:.2f}")
                subtot += quant*cost
            print(f"{self.name} spent ${subtot:.2f} on {name}s.")
            total += subtot
        print(f"\nTOTAL: ${total:.2f}")
```

Python Essentials: Advanced Structures

- Object-Oriented

```
class shopCart:
    def __init__(self, customer_name):
        self.name = customer_name
        self.cart = {}
    def add_item(self, name, cost, quantity):
        if(name in self.cart.keys()):
            self.cart[name].append([cost, quantity])
        else:
            self.cart[name] = [[cost, quantity]]
    def print_cart(self):
        total = 0.0
        for name, prices in self.cart.items():
            subtot = 0.0
            print("\n"+name.upper()+"S:")
            print("\tUnits\tCost\tSubtotal")
            for (cost, quant) in prices:
                print(f"\t{quant:d}\t${cost:.2f}\t${quant*cost:.2f}")
                subtot += quant*cost
            print(f"{self.name} spent ${subtot:.2f} on {name}s.")
            total += subtot
        print(f"\nTOTAL: ${total:.2f}")
```

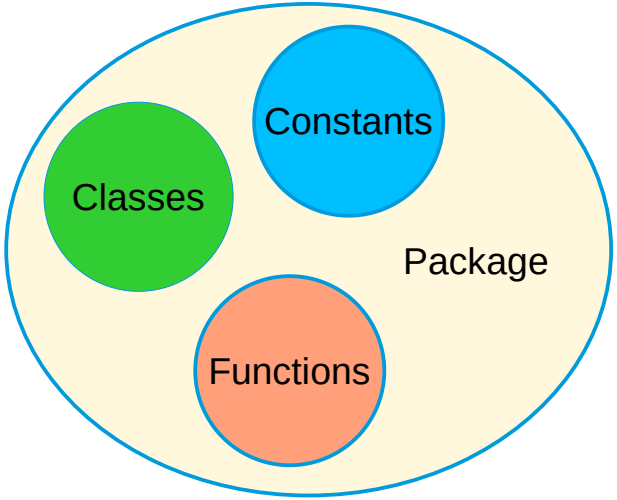
shop.py

```
if __name__ == "__main__":
    myCart = shopCart("Josh")
    myCart.add_item("apple", 0.42, 5)
    myCart.add_item("apple", 0.44, 2)
    myCart.add_item("grape", 0.02, 10)
    myCart.add_item("grape", 0.01, 100)
    myCart.add_item("grape", 0.24, 4)
    myCart.add_item("orange", 5.42, 1)
    myCart.print_cart()
```

```
from shop import shopCart
myCart = shopCart("Josh")
myCart.add_item("apple", 0.42, 5)
myCart.add_item("apple", 0.44, 2)
myCart.add_item("grape", 0.02, 10)
myCart.add_item("grape", 0.01, 100)
myCart.add_item("grape", 0.24, 4)
myCart.add_item("orange", 5.42, 1)
myCart.print_cart()
```

Python Essentials: Advanced Structures

- Packages



The Standard Library

Python Essentials: Tutorial & Challenge

- **Challenge**

Write a class that creates a collection of students' names, areas of studies, and course grades.