Introduction to Big Data and Data Science (CSCE 5300 Section 005)*

Yunhe Feng

Assistant Professor, Department of Computer Science and Engineering

29th August, 2024

*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)

1 Introduction to Python Programming

2 Data Structures

3 Object-oriented Programming (OOP)

4 Assignment

Introduction to Python Programming

- What is Python
- Why Python
- Python VS C++

What is Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- 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.

Products and Tools Written in Python

- Web Applications
  - **Instagram**: One of the largest social media platforms started as a Python project.
  - **Pinterest**: A popular image-sharing platform built using the Django web framework.
- Desktop Applications
  - **Dropbox**: The desktop client was initially written in Python.
- Scientific and Numeric Applications
  - **SciPy**: An open-source library used for high-level computations.
  - **Pandas**: A powerful tool for data analysis and manipulation.
- Security Tools
  - **SQLMap**: An open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.

Top Programming Languages (Aug. 29, 2023)

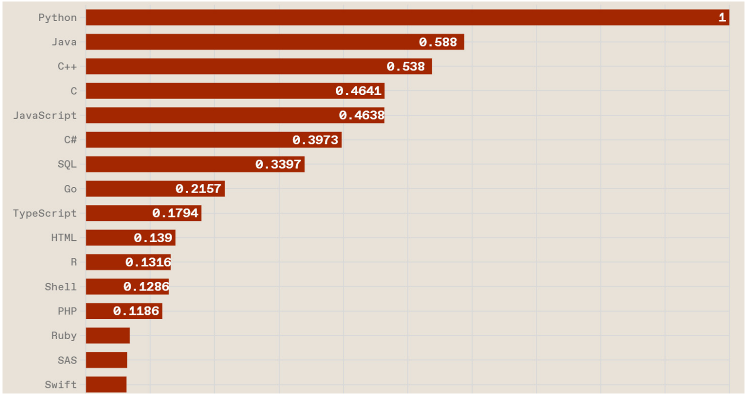
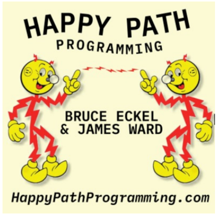


Figure 1: IEEE Spectrum's 10th annual rankings of the Top Programming Languages[1]

[1] https://spectrum.ieee.org/the-top-programming-languages-2023

Why Python

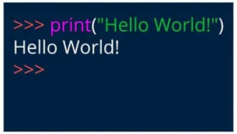
Life is short (You need Python) - Bruce Eckel



- The author of **Thinking in Java** and **Think in Python**
- One of the two anchors of `HappyPathProgramming.com`

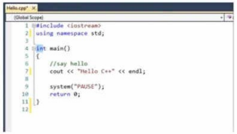
Why Python - Hello World Program

Python



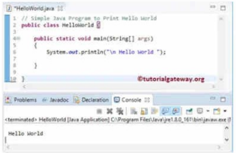
```
>>> print("Hello World!")
Hello World!
>>>
```

C++



```
#include <iostream>
using namespace std;

int main()
{
    //say hello
    cout << "Hello C++" << endl;

    system("PAUSE");
    return 0;
}
```

Java



```
// Simple Java Program to Print Hello World
public class HelloWorld {

    public static void main(String[] args)
    {
        System.out.println("\n Hello World ");

    }

}
```

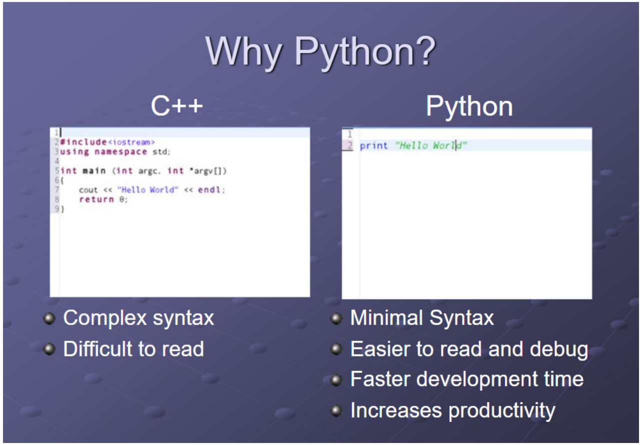
```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe

Hello World
```

Assembly

Why Python

- Python is easy to learn and use
- Python is flexible
- Python has an active, supportive community
- Hundreds of Python libraries and frameworks
- Python is well suited to data science, machine learning, and AI

Python VS C++



Why Python?

C++

```
#include<iostream>
using namespace std;

int main (int argc, int *argv[])
{
    cout << "Hello World" << endl;
    return 0;
}
```

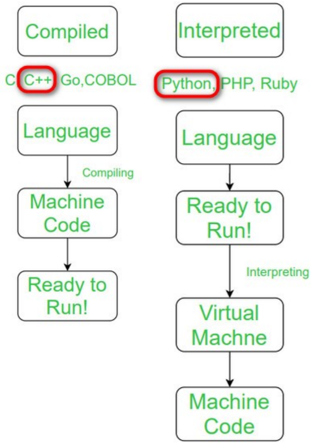
- Complex syntax
- Difficult to read

Python

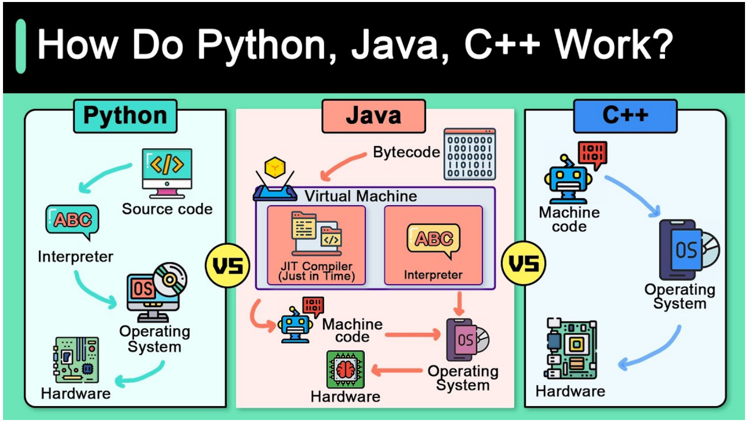
```

print "Hello World"
```

- Minimal Syntax
- Easier to read and debug
- Faster development time
- Increases productivity

Python VS C++


Compiled
C, C++, Go,COBOL
Language
Compiling
Machine Code
Ready to Run!
Interpreted
Python, PHP, Ruby
Language
Ready to Run!
Interpreting
Virtual Machne
Machine Code

Python, Java, C++



1 Introduction to Python Programming

2 Data Structures

3 Object-oriented Programming (OOP)

4 Assignment

Data Structures - Outcomes

- Become familiar with data structures in Python
- Familiarize yourself with String, Lists, and Dictionaries
- Python programming tools for data management
- Apply the knowledge in processing, analyzing, and modifying datasets for targeted outcomes

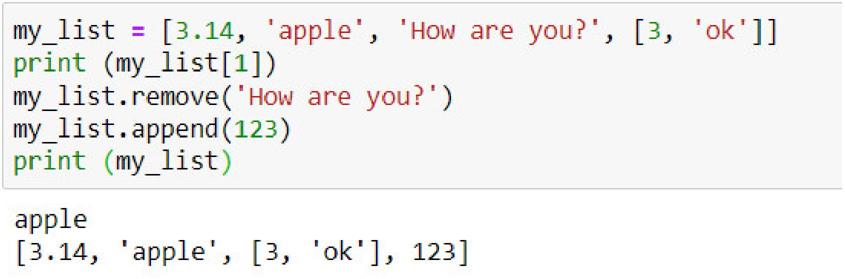
Basic Data Structures

- Int
- Float
- Boolean
- String
- List
- Dictionary
- Set
- Tuple

Lists: Ordered Sequences of Data

- Elements: numbers, strings, lists of lists, etc.
- Operations: Add, remove, change/modify
- Example
  - my_list = [3.14, ‘apple’, ‘How are you?’, [3, ‘ok’]]
- Try the following and see what happens to the my_list? (Exam question)
  - print (my_list[1])
  - my_list.remove(’How are you?’)
  - my_list.append(123)
  - print (my_list)

Lists: Demo Result

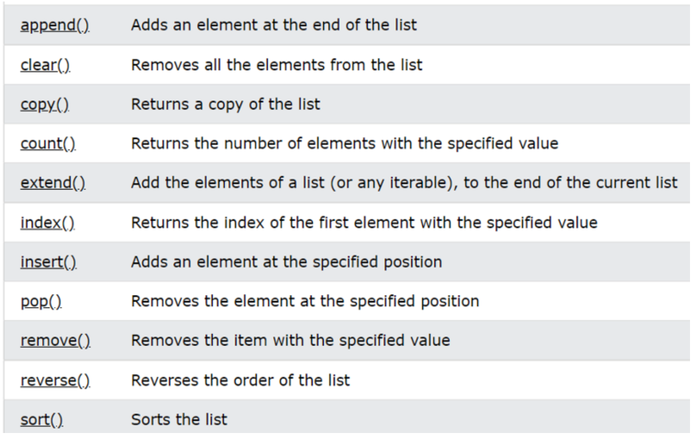


```
my_list = [3.14, 'apple', 'How are you?', [3, 'ok']]
print (my_list[1])
my_list.remove('How are you?')
my_list.append(123)
print (my_list)
```

```
apple
[3.14, 'apple', [3, 'ok'], 123]
```

Figure 2: Demo result

Operations on List



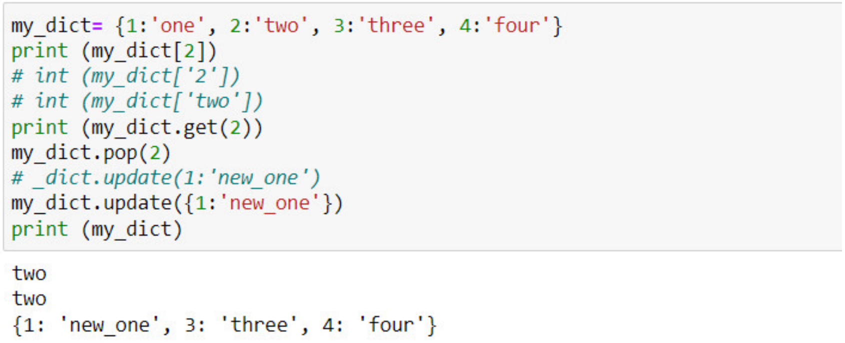
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Figure 3: Operations on the list

Dictionary: Store Items in ‘pairs’ {key: value}

- Elements: numbers, strings, lists of lists, etc.
- Operations: Add, remove, change/modify
- Example
  - my_dict = {1:’one’, 2:’two’, 3:’three’, 4:’four’}
- Try the following and see what happens to the my_dict ?
  - print (my_dict[2])
  - print (my_dict[‘2’])
  - print (my_dict[‘two’])
  - print (my_dict.get(2))
  - my_dict.pop(2)
  - my_dict.update(1:‘new_one’)
  - my_dict.update({1:‘new_one’})

Dictionary: Demo Result

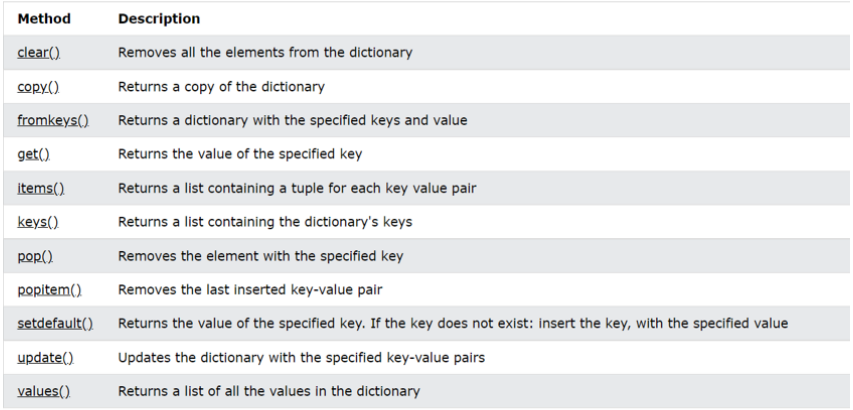


```
my_dict= {1:'one', 2:'two', 3:'three', 4:'four'}
print (my_dict[2])
# int (my_dict['2'])
# int (my_dict['two'])
print (my_dict.get(2))
my_dict.pop(2)
# _dict.update(1:'new_one')
my_dict.update({1:'new_one'})
print (my_dict)
```

```
two
two
{1: 'new_one', 3: 'three', 4: 'four'}
```

Figure 4: Demo result

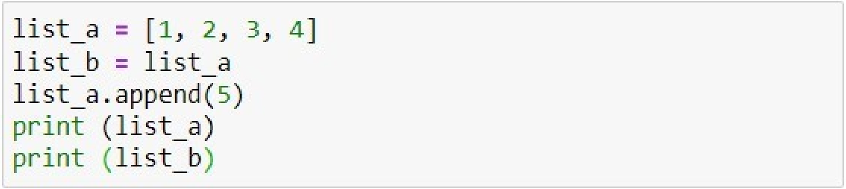
Operations on Dictionary



Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

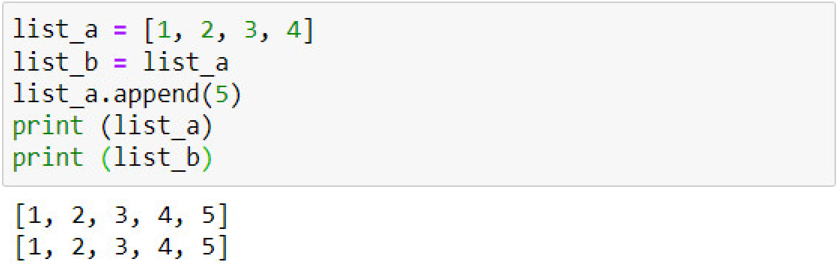
Figure 5: Operations on the dictionary

Copy Mutable Objects by Reference - List



```
list_a = [1, 2, 3, 4]
list_b = list_a
list_a.append(5)
print (list_a)
print (list_b)
```

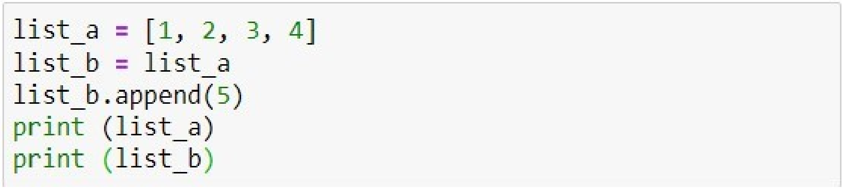
Copy Mutable Objects by Reference - List



```
list_a = [1, 2, 3, 4]
list_b = list_a
list_a.append(5)
print (list_a)
print (list_b)
```

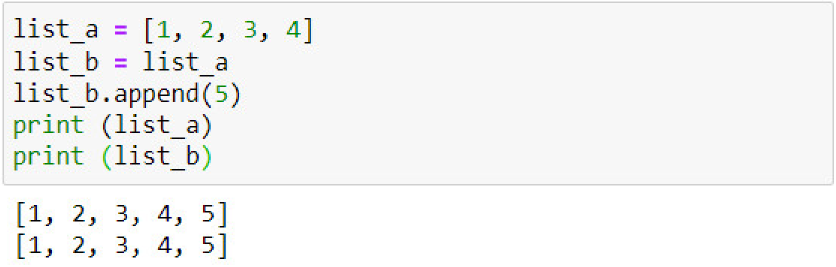
```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Copy Mutable Objects by Reference - List



```
list_a = [1, 2, 3, 4]
list_b = list_a
list_b.append(5)
print (list_a)
print (list_b)
```

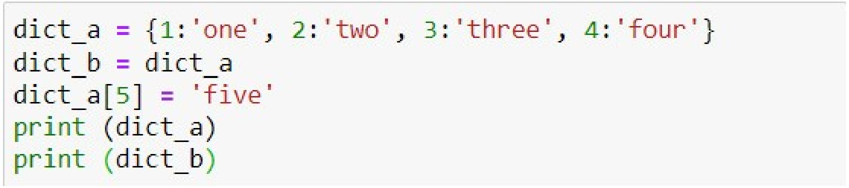
Copy Mutable Objects by Reference - List



```
list_a = [1, 2, 3, 4]
list_b = list_a
list_b.append(5)
print (list_a)
print (list_b)
```

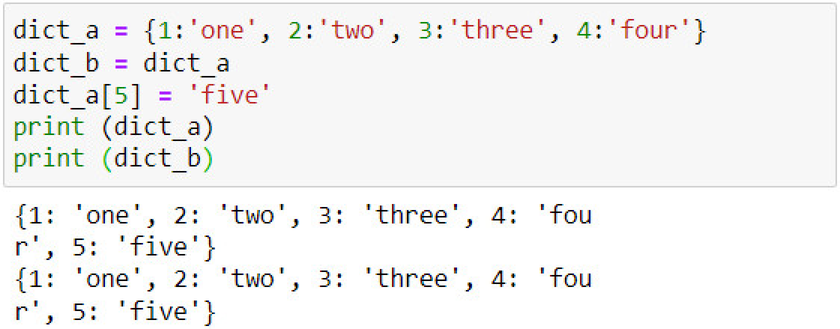
```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Copy Mutable Objects by Reference - Dictionary



```
dict_a = {1:'one', 2:'two', 3:'three', 4:'four'}
dict_b = dict_a
dict_a[5] = 'five'
print (dict_a)
print (dict_b)
```

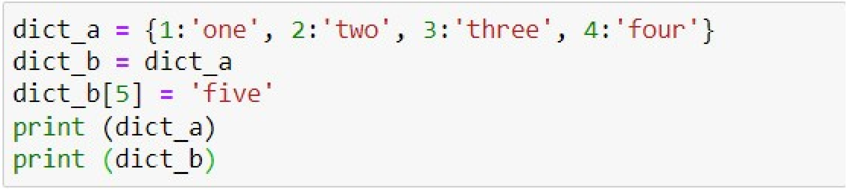
Copy Mutable Objects by Reference - Dictionary



```
dict_a = {1:'one', 2:'two', 3:'three', 4:'four'}
dict_b = dict_a
dict_a[5] = 'five'
print (dict_a)
print (dict_b)
```

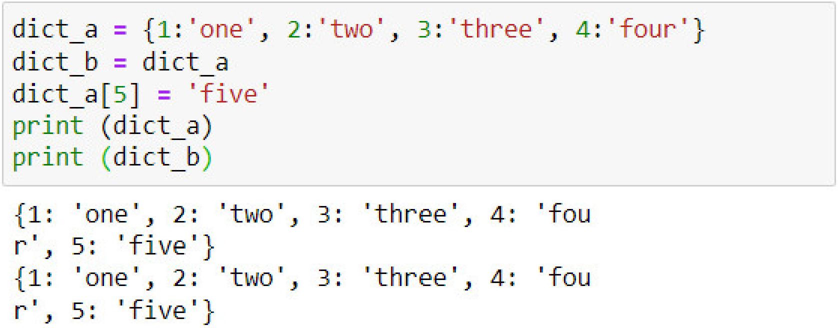
```
{1: 'one', 2: 'two', 3: 'three', 4: 'fou
r', 5: 'five'}
{1: 'one', 2: 'two', 3: 'three', 4: 'fou
r', 5: 'five'}
```

Copy Mutable Objects by Reference - Dictionary



```
dict_a = {1:'one', 2:'two', 3:'three', 4:'four'}
dict_b = dict_a
dict_b[5] = 'five'
print (dict_a)
print (dict_b)
```

Copy Mutable Objects by Reference - Dictionary

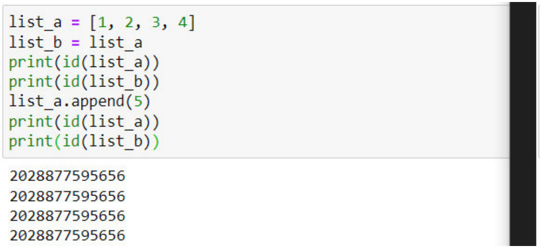


```python
dict_a = {1:'one', 2:'two', 3:'three', 4:'four'}
dict_b = dict_a
dict_a[5] = 'five'
print (dict_a)
print (dict_b)
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'fou
r', 5: 'five'}
{1: 'one', 2: 'two', 3: 'three', 4: 'fou
r', 5: 'five'}
```

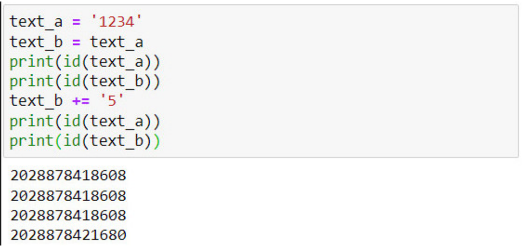
Copy Mutable Objects by Reference

id(object) returns the identity of an object



```
list_a = [1, 2, 3, 4]
list_b = list_a
print(id(list_a))
print(id(list_b))
list_a.append(5)
print(id(list_a))
print(id(list_b))
```

```
2028877595656
2028877595656
2028877595656
2028877595656
```



```
text_a = '1234'
text_b = text_a
print(id(text_a))
print(id(text_b))
text_b += '5'
print(id(text_a))
print(id(text_b))
```

```
2028878418608
2028878418608
2028878418608
2028878421680
```

Mutable Objects and Immutable Obejects in Python

- Mutable Objects
  - list
  - dictionary
  - set
  - user-defined classes
- Immutable Objects
  - int
  - float
  - bool
  - string
  - tuple
  - range

1 Introduction to Python Programming

2 Data Structures

3 Object-oriented Programming (OOP)

4 Assignment

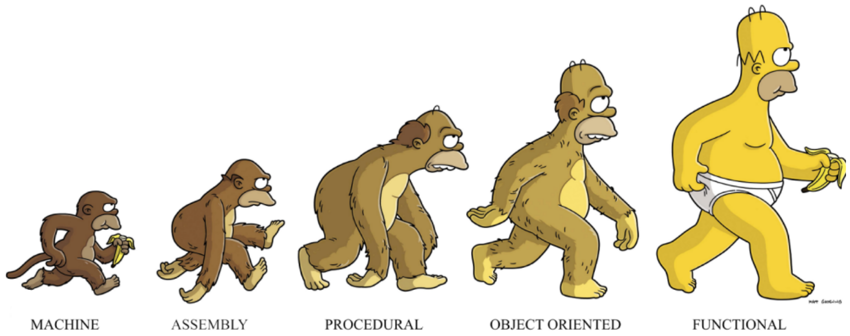
Object-oriented Programming (OOP)

- What is OOP
- Why OOP

What is OOP

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior. Properties and behaviors are bundled into individual objects.

Programming Evolution[1]



[1] www.shorturl.at/bflOQ

OOP vs Procedural vs Functional Programming[2]

- **Object-oriented** code helps the programmer think of objects which represent a concept or real-world component
- **Procedural** programming organizes the code into chunks of procedures
- **Functional** programming orients the programmer in the world of pure functions

[2] `https://www.scaler.com/topics/java/oop-vs-functional-vs-procedural/`

OOP vs Procedural vs Functional Programming[3]

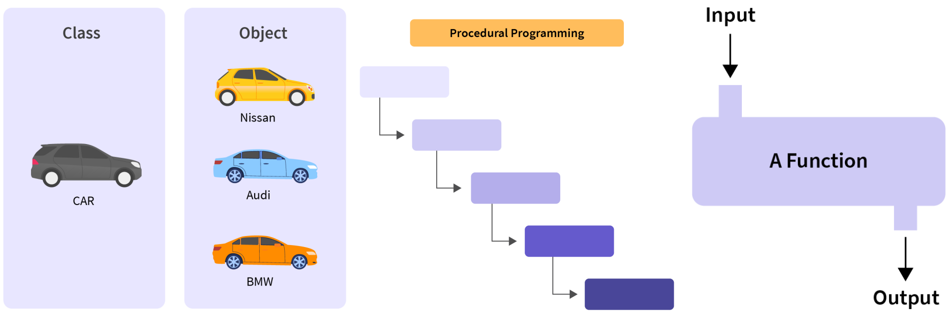


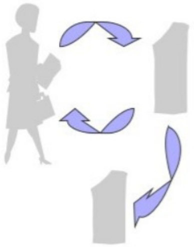
Figure 6: OOP vs Procedural vs Functional programming

[3] https://www.scaler.com/topics/java/oop-vs-functional-vs-procedural/

OOP vs Procedural Programming

Procedural vs. Object-Oriented

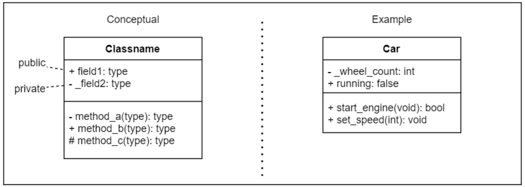
- Procedural
- Object Oriented

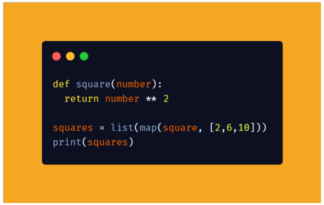


Withdraw, deposit, transfer

Customer, money, account

OOP vs Functional Programming[4],[5]

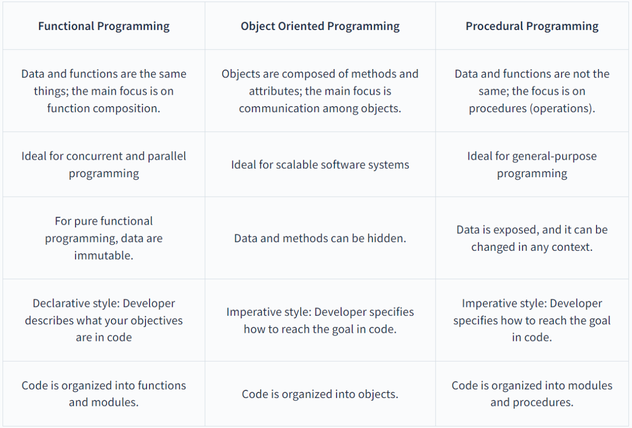




```python
def square(number):
  return number ** 2

squares = list(map(square, [2,6,10]))
print(squares)
```

[4]https://aswinbarath.github.io/functional-programming-in-python/
[5]https://sbcode.net/python/uml_diagrams/

Difference between Functional vs OOP vs Procedural Programming[6]



Functional Programming	Object Oriented Programming	Procedural Programming
Data and functions are the same things; the main focus is on function composition.	Objects are composed of methods and attributes; the main focus is communication among objects.	Data and functions are not the same; the focus is on procedures (operations).
Ideal for concurrent and parallel programming	Ideal for scalable software systems	Ideal for general-purpose programming
For pure functional programming, data are immutable.	Data and methods can be hidden.	Data is exposed, and it can be changed in any context.
Declarative style: Developer describes what your objectives are in code	Imperative style: Developer specifies how to reach the goal in code.	Imperative style: Developer specifies how to reach the goal in code.
Code is organized into functions and modules.	Code is organized into objects.	Code is organized into modules and procedures.

[6] https://www.scaler.com/topics/java/oop-vs-functional-vs-procedural/

OOP in Python

https://realpython.com/python3-object-oriented-programming/

1 Introduction to Python Programming

2 Data Structures

3 Object-oriented Programming (OOP)

4 Assignment

Assignment-1 (4.0 pt.)

- Basic Python Syntax (2 pts.)
- Object Oriented Programming (2 pts.)
- Due time: 11:59:59 PM, September 4, 2024 (End of next Wednesday)
- The assignment and how to submit your assignment will be announced on Canvas

*The teaching materials are reorganized and reformed based on Prof. Ravi Vadapalli's slides (Ravi.Vadapalli@unt.edu, UNT & University of Miami)