Introduction to Programming

PYTHON

Chapter 2 – Concepts and Forms of Data

Presenter: **Dr. Nguyen Dinh Long**

Email: dinhlonghcmut@gmail.com

Google-site: https://sites.google.com/view/long-dinh-nguyen

Oct. 2022

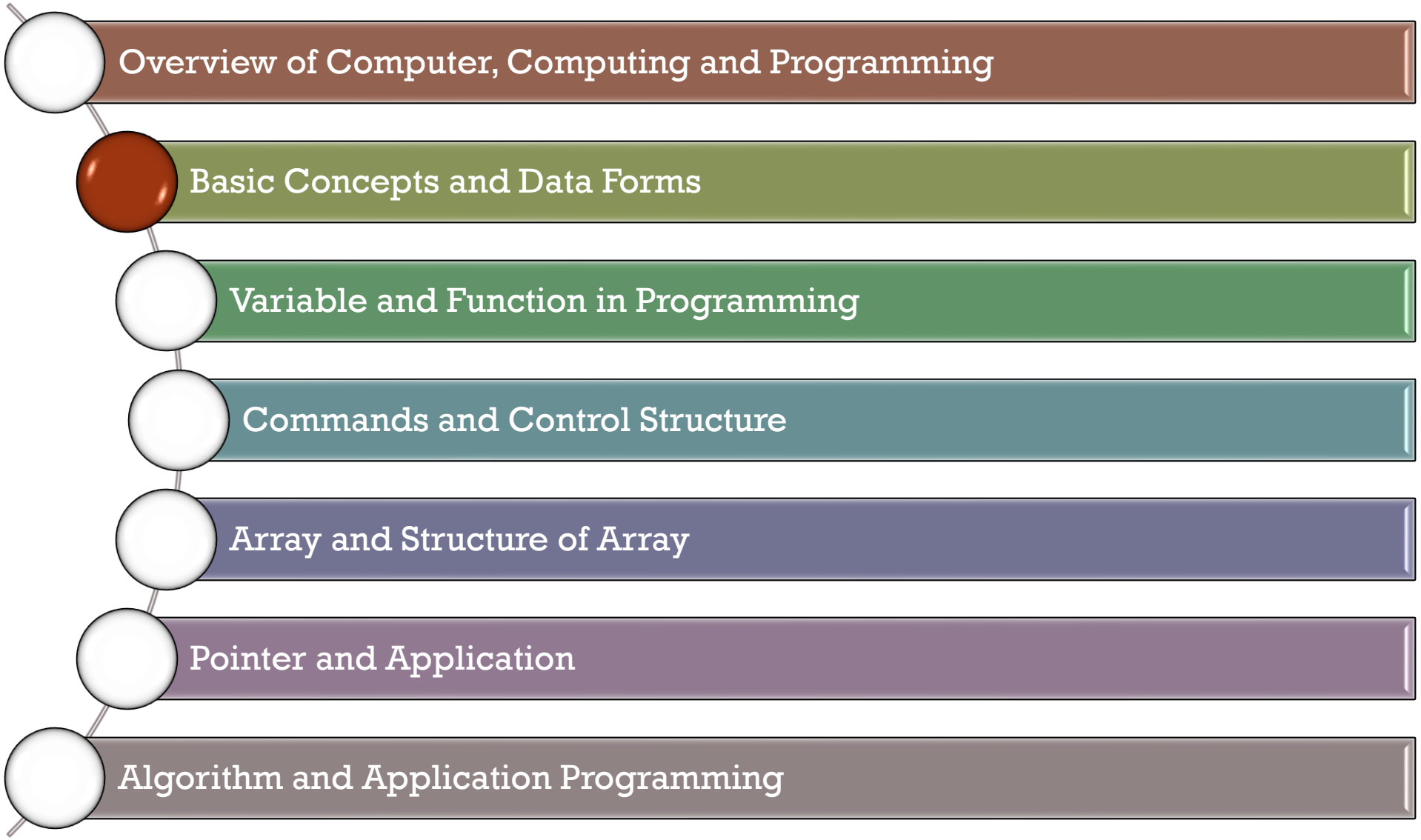
Programming

Hamilton was a self-taught programmer, working in the US in the 1960's. Owing to the success of her previous work, Hamilton was the first programmer to be hired for the Apollo project. She became the Director of Software Engineering at the MIT Instrumentation lab. Her lab developed the on-board flight software for NASA's Apollo space project, which took humankind to the moon.

The achievement was a monumental task at a time when computer technology was in its infancy: The astronauts had access to only 72 kilobytes of computer memory (a 256-gigabyte cell phone today carries almost a million times more storage space). Programmers had to use paper punch cards to feed information into room-sized computers with no screen interface.

Margaret Hamilton, NASA's lead software engineer for the Apollo, stands next to the code she wrote by hand that took humanity to the moon in 1969.

Outline


Overview of Computer, Computing and Programming
Basic Concepts and Data Forms
Variable and Function in Programming
Commands and Control Structure
Array and Structure of Array
Pointer and Application
Algorithm and Application Programming

References

Main:

- Maurizio Gabbrielli and Simone Martini, 2010. *Programming Languages: Principles and Paradigms*, Springer.
- Cao Hoàng Trụ, 2004. *Ngôn ngữ lập trình- Các nguyên lý và mô hình*, Nhà xuất bản Đại học Quốc gia Tp. Hồ Chí Minh

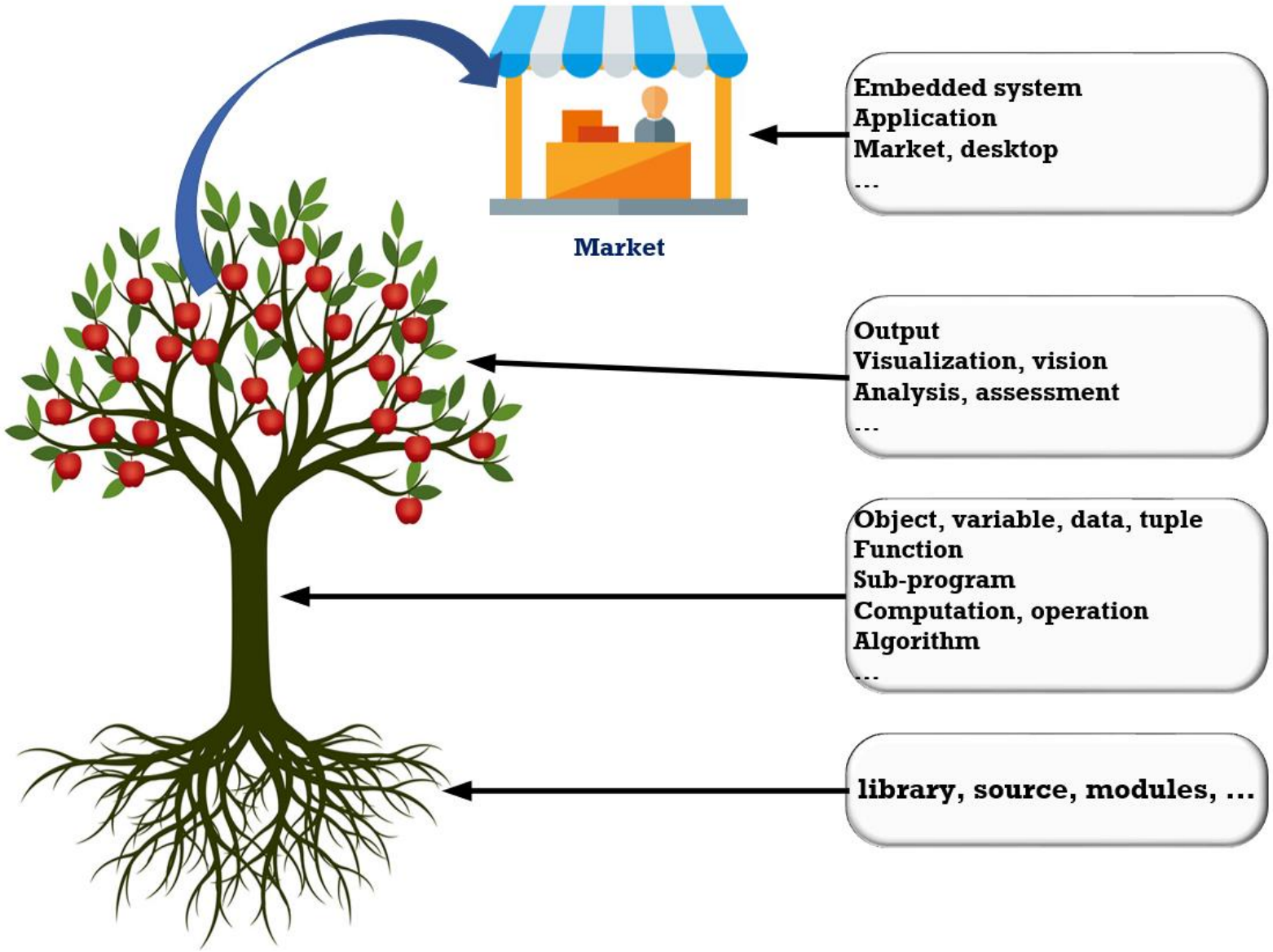
More:

- Wes McKinney, 2013. *Python for Data Analysis*, O'Reilly Media.
- Guido van Rossum, Fred L. Drake, Jr.,, 2012. *The Python Library Reference*, Release 3.2.3.

- Slides here are collected and modified from several sources in Universities and Internet.

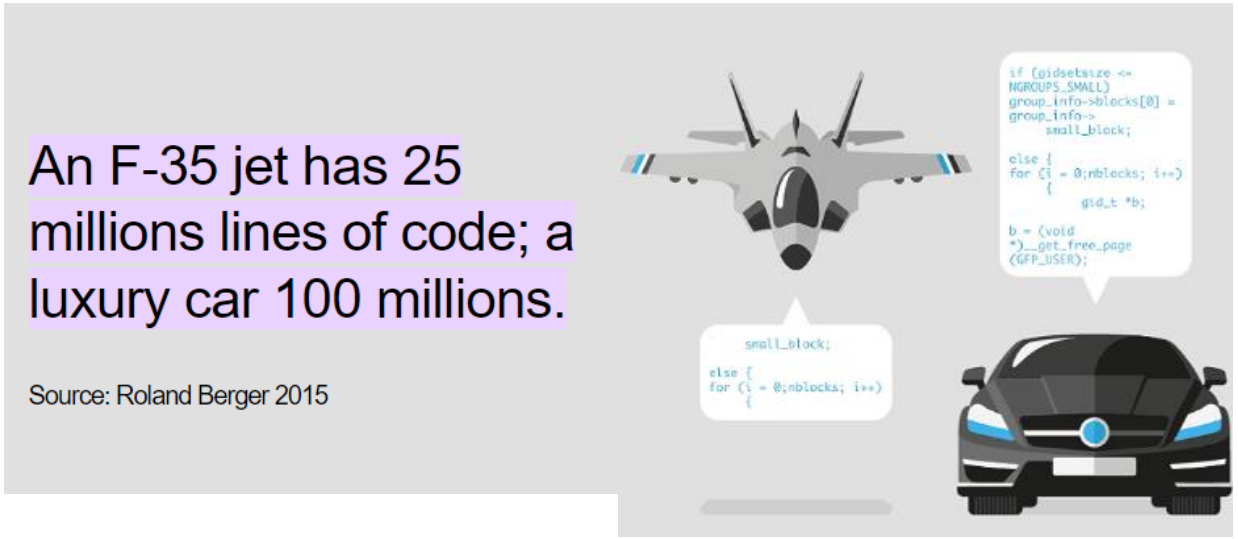
Computer programs

- **General structure:**



Computer programs

- **Thinking ...**



62 million lines

Not including backend code,
Facebook runs on 62 million lines of code.

Microsoft Windows—one of the most complex software tools ever built for a single computer—is about 50 million lines.

Google Is 2 Billion Lines of Code—And It's All in One Place

Content of Chapter 2

1. Structure of Computer programs
   Python for Data Analysis

1. Examples of programs

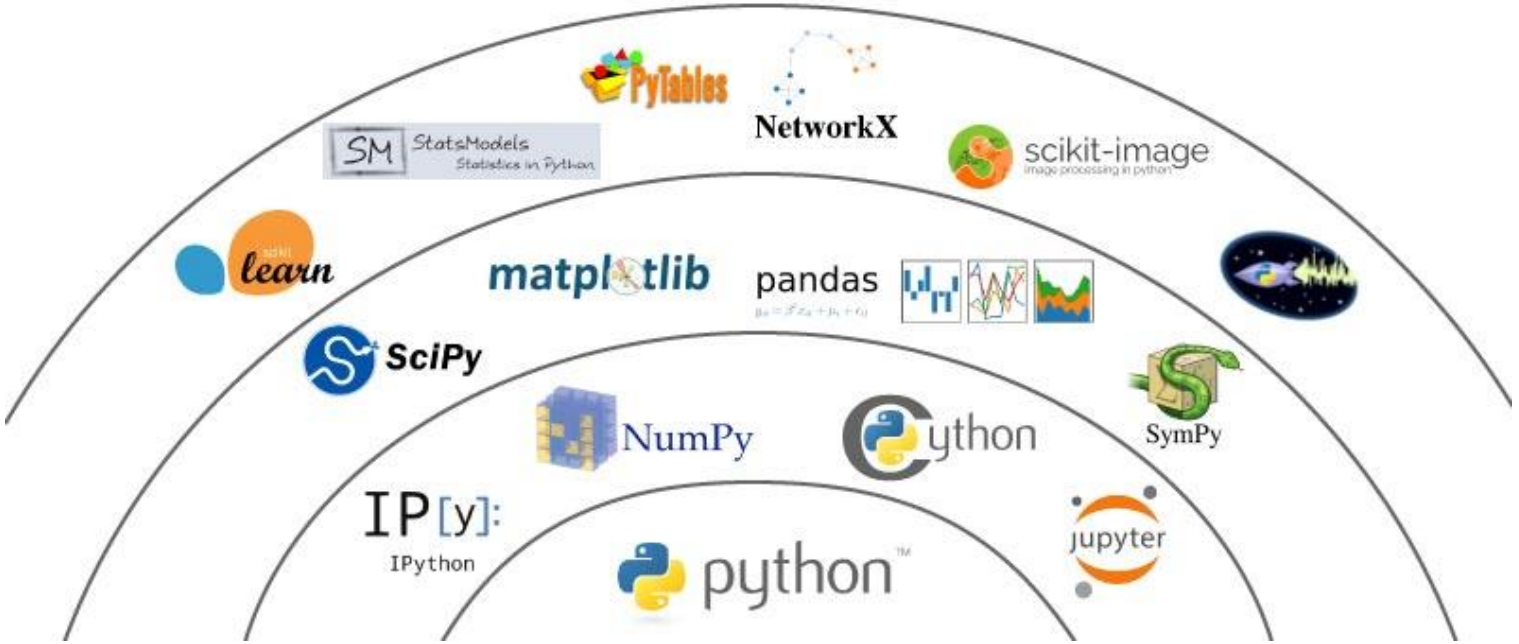
2. Data types and computation

3. Python library for programming

4. Examples of using Python library and Applications

Programming Library (for Python)

- A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs.
- It makes Python Programming simpler and convenient for the programmer.
- Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, ...
- We use libraries so that we don’t need to write the code again in our program that is already available.
- The Python standard library consists of more than 200 core modules. All these work together to make Python a high-level programming language. Python Standard Library plays a very important role.



Programming Library (for Python)

TensorFlow:

- This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.

Matplotlib:

- This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

Pandas:

- Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

Programming Library (for Python)

❑ Numpy:

- The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.

❑ SciPy:

- The name “SciPy” stands for “Scientific Python”. It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

❑ Scrapy:

- It is an open-source library that is used for extracting data from websites. It provides very fast web crawling and high-level screen scraping. It can also be used for data mining and automated testing of data.

Programming Library (for Python)

❑ Scikit-learn:

- It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

❑ PyGame:

- This library provides an easy interface to the Standard Directmedia Library (SDL) platform-independent graphics, audio, and input libraries. It is used for developing video games using computer graphics and audio libraries along with Python programming language.

❑ PyTorch:

- PyTorch is the largest machine learning library that optimizes tensor computations. It has rich APIs to perform tensor computations with strong GPU acceleration. It also helps to solve application issues related to neural networks.

Programming Library (for Python)

- **PyBrain:**

- The name “PyBrain” stands for Python Based Reinforcement Learning, Artificial Intelligence, and Neural Networks library. It is an open-source library built for beginners in the field of Machine Learning. It provides fast and easy-to-use algorithms for machine learning tasks. It is so flexible and easily understandable and that’s why is really helpful for developers that are new in research fields.

As we write large-size programs in Python, we want to maintain the code’s modularity. We split the code into different parts and we can use that code later ever we need it.

In Python, modules play that part. Instead of using the same code in different programs and making the code complex, we define mostly used functions in modules and we can just simply import them in a program wherever there is a requirement.

We don’t need to write that code but still, we can use its functionality by importing its module. Multiple interrelated modules are stored in a library. And whenever we need to use a module, we import it from its library.

In Python, it’s a very simple job to do due to its easy syntax.

Programming Library (for Python)

❑ **Example:**

- Import “math”

```
# Importing math library
import math

A = 16
print(math.sqrt(A))
```

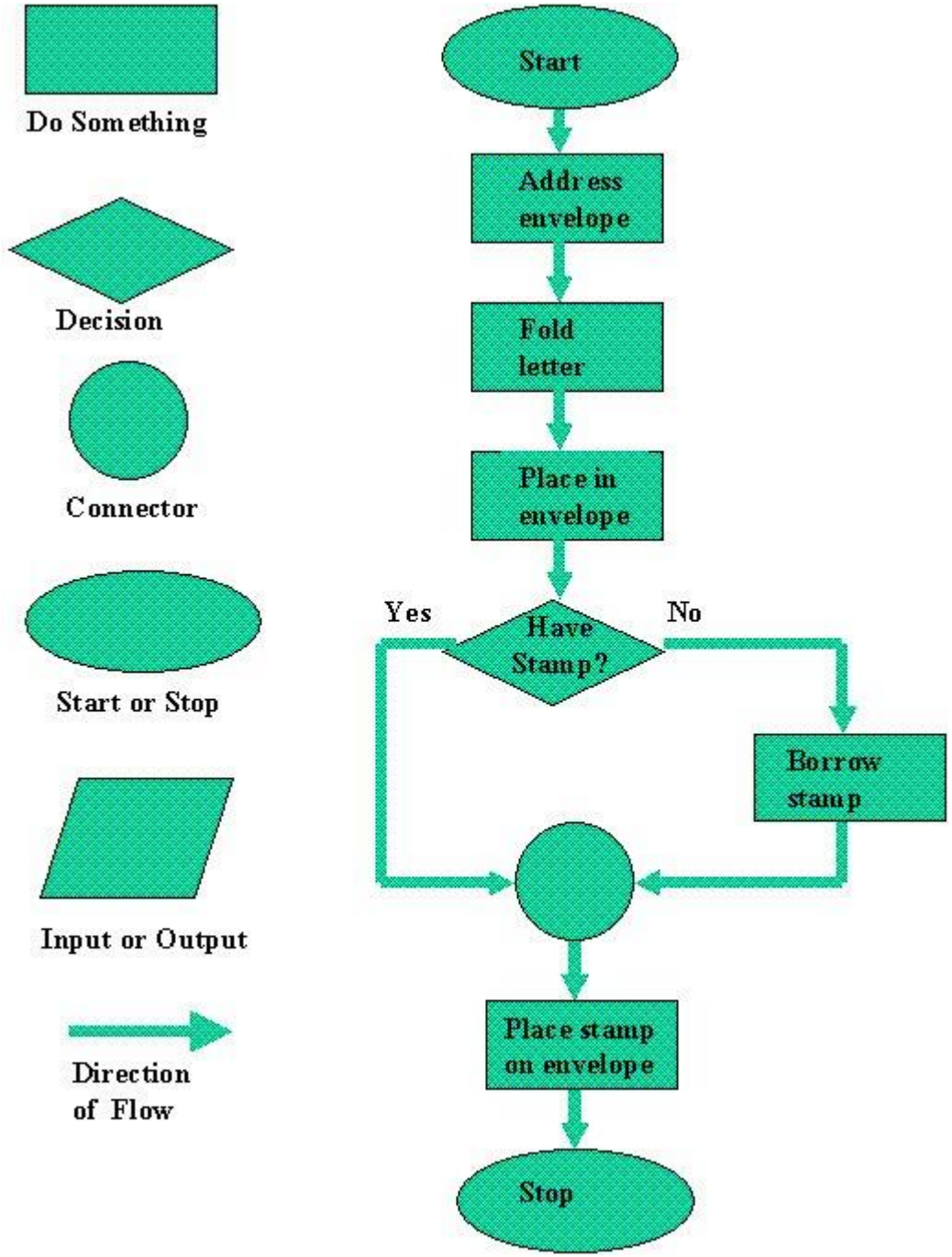
- Import specific item in a library module (saving resource)

```
# Importing specific items
from math import sqrt, sin

A = 16
B = 3.14
print(sqrt(A))
print(sin(B))
```

Structure of Computer programs

- **Computer programming:**
  - Objects
  - Types
  - Variables
  - Methods
  - Tuples



Structure of Computer programs

❑ Objects:

- Programs work with data. Every piece of data in a Python program is called an object.
- Objects can be very small (the number 3, 4, 10) or very large (a digital photograph, video …).
- Every object has a type. The type determines what you can do with an object.

- The PythonZoo:
  - Imagine there is a zoo inside your Python interpreter.
  - Every time you create an object, an animal isborn.
  - What an animal can do depends on the type (kind) of animal: birds can fly, fish can swim, elephants can lift weights, ...
  - When an animal is no longer used, it dies (disappears).

Structure of Computer programs

- **Making Objects:**

- Create objects as follows:

Numbers: Simply write them:

```
13
3.14159265
-5
3 + 6j
```

Complex number

Strings: (a piece of text)
Write text between quotation marks ("and 'are both okay):

```
"CS101 is wonderful"
'The instructor said: "Well done!" and smiled'
```

Booleans: (truth values)
Write `True` or `False`.

Structure of Computer programs

Making Objects:

- Complicated objects are made by calling functions that create them:

```
from cs1robots import *
Robot()

from cs1media import *
load_picture("photos/geowi.jpg")
```

- A tuple object is an object that contains other objects. To create a tuple, write objects separated by commas

```
(3, 2.5, 7)
("red", "yellow", "green")
(20100001, "Hong Gildong")
```

Structure of Computer programs

- **Types:**

- Every object has a type. The type determines what the object can do, and what you can do with the object. For instance, you can add two numbers, but you cannot add two robots.

```
>>> type(3)
<class 'int'>
>>> type(3.1415)
<class 'float'>
>>> type("CS101 is fantastic")
<class 'str'>
>>> type(3 + 7j)
<class 'complex'>
>>> type(True)
<class 'bool'>
```

Integer number: **int**

Floating point number: **float**

String: **str**

Complex number: **complex**

Boolean: **bool**

Structure of Computer programs

- **Object Names (Variables):**

- Objects can be given a name:

```
message = "CS101 is fantastic"

n = 17
hubo = Robot()
pi = 3.1415926535897931
finished = True
img = load_picture("geowi.jpg")
```

- In the Python zoo, the name is a sign board on the animal's cage.

Structure of Computer programs

❑ **Variables:**

- Names are often called variables, because the meaning of a name is variable: the same name can be assigned to different objects within a program.

```
n = 17
n = "Seventeen"
n = 17.0
```

- The object assigned to a name is called the value of the variable. The value can be changed over time.

- To indicate that a variable is empty, we use the special object None (of class 'NoneType'):

```
n = None
```

Structure of Computer programs

❑ **Name structure of Variables:**

- The rules for variable and function names:
  - A name consists of letters, digits, and the underscore _.
  - The first character of a name should not be a digit.
  - The name should not be a keyword such as def, if, else, or while.
  - Upper case and lower case are different: Pi is not the same as pi.

Good:

```
my_message = "CS101 is fantastic"
a13 = 13.0
```

Bad:

```
more@ = "illegal character"
13a = 13.0
def = "Definition 1"
```

Structure of Computer programs

- **Methods, Algorithms:**

- What objects can do depends on the type of object: a bird can fly, a fish can swim.
- Objects provide methods to perform these actions.
- The methods of an object are used through dot-syntax:

```
>>> hubo = Robot()
>>> hubo.move()
>>> hubo.turn_left()

>>> img = load_picture("geowi.jpg")
>>> print(img.size())  # width and height in pixels
(58, 50)
>>> img.show()  # display the image

>>> b = "banana"
>>> print(b.upper())
```

Structure of Computer programs

❑ **Methods:**

- Operators: for numbers, we use the operators +, -, *, /, //,%, and **
- Expressions: a combination of objects, variables, operators, and function calls:

```
3.0 * (2 ** 15 - 12 / 4) + 4 ** 3
```

  - The operators have precedence as in mathematics:
    - 1. exponentiation **
    - 2. multiplication and division *, /, //,%
    - 3. addition and subtraction +,-

❖ When in doubt, use parentheses

❖ All operators also work for complex numbers

Structure of Computer programs

❑ **Methods:**

- Expressions: a combination of objects, variables, operators, and function calls:
  - String expressions

```
>>> "Hello" + "CS101"
'HelloCS101'

>>> "CS101 " * 8
'CS101 CS101 CS101 CS101 CS101 CS101 CS101 CS101 '
```

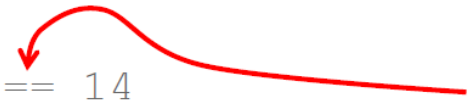
  - Boolean expressions

A Boolean expression is an expression whose value has type bool. They are used in if and while statements.

The operators ==, !=, >, <, <=and >=return boolean values.

```
>>> 3 < 5
True
>>> 3.14 >= 3.14
True
>>> "Cheong" < "Choe"
True
```

```
>>> 27 == 14
False
>>> 3.14 != 3.14
False
>>> "3" == 3
False
```



Structure of Computer programs

Tuples:

- A tuple is an object that contains other objects:

```
>>> position = (3.14, -5, 7.5)
>>> profs = ("Ko", "Kim", "Baik", "Song", "Lee")
```

- A tuple is a single object of type tuple:

```
>>> print(position, type(position))
(3.14, -5, 7.5) <class 'tuple'>
```

- Packing and unpacking tuples

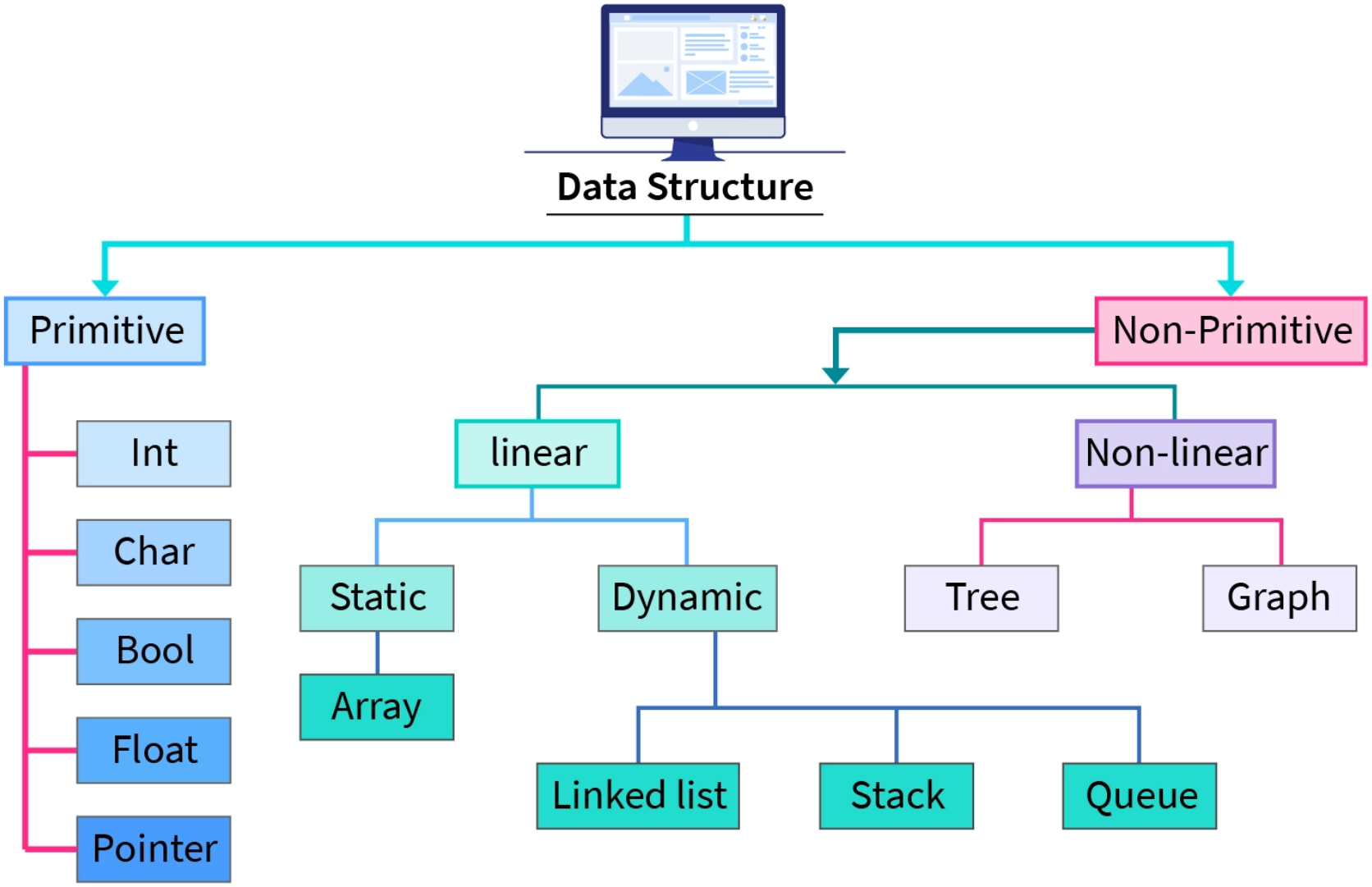
```
>>> x, y, z = position
>>> print(x)
```

```
>>> a, b = ("aa", "bb")
>>> a, b = b, a
>>> print(b)
```

Data types for Programming

- **Data structure:**

.



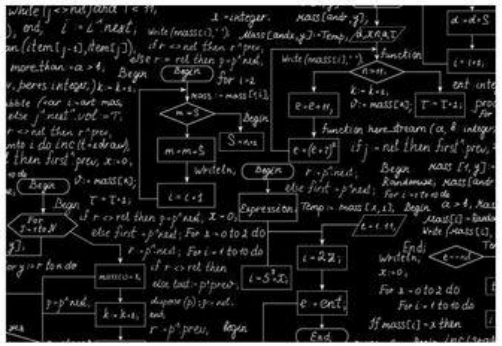
Data types for Programming

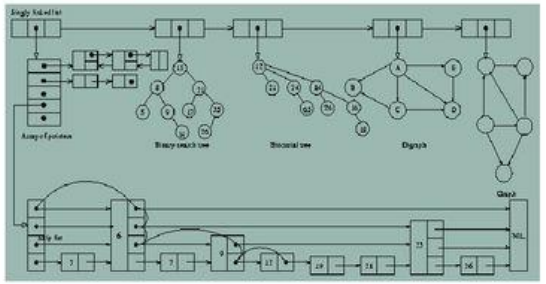
- **Data structure:**

.

Why Data Structures?

Algorithms + Data Structures = Programs
-Niklaus Wirth





Data types for Programming

Primitive Data Types:

- Booleans

The type of logical values, or booleans, is composed of:

• Values: The two truth values, true and false.

• Operations: an appropriate selection from the main logical operations: conjunction (and), disjunction (or) and negation (not), equality, exclusive or, …

- Reals

The so-called real type (or floating point numbers) is composed of:

• Values: an appropriate subset of the rational numbers, usually fixed when the language is defined (but there are case in which it is fixed by the specific abstract machine, a matter that deeply affects portability); the structure (size, granularity, etc.) of such a subset depends on the representation adopted.

• Operations: comparisons and an appropriate selection of the main numeric operations (addition, subtraction, multiplication, division, exponentiation, square roots, etc.)

Data types for Programming

Primitive Data Types:

- Integer

The most common primitive numeric data type is integer. The hardware of many computers supports several sizes of integers. These sizes of integers, and often a few others, are supported by some programming languages. For example, four signed integer sizes: byte, short, int, and long.

The type of integer numbers is composed of:

• Values: A finite subset of the integers, usually fixed when the language is defined (but there are cases in which it is determined by the abstract machine which can be the cause of some portability problems).

Due to representation issues, the interval $[-2^t , 2^t - 1]$ is commonly used.

• Operations: the comparisons and an appropriate selection of the main arithmetic operators (addition, subtraction„ multiplication, integer division, remainder after division, exponentiation, ...).

Data types for Programming

❑ Primitive Data Types:

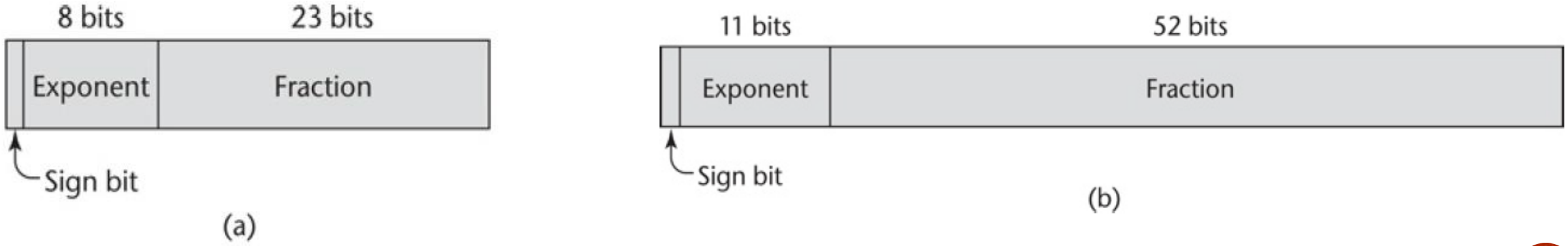
- Fixed Point

• Values: an appropriate subset of the rational numbers, usually fixed when the language is defined; the structure (size, granularity, etc.) of such a subset depends on the representation adopted.

• Operations: comparisons and an appropriate selection of the main numeric operations (addition, subtraction, multiplication, division, exponentiation, extraction of square roots, etc.).

- Floating-Point

Floating-point data types model real numbers, but the representations are only approximations for many real values. For example, neither of the fundamental numbers π or e (the base for the natural logarithms) can be correctly represented in floating-point notation.



IEEE Floating-Point Standard 754 format for single and double-precision representation

Data types for Programming

Primitive Data Types:

- Complex

Some programming languages support a complex data type—for example, Python. Complex values are represented as ordered pairs of floating-point values.

In Python, the imaginary part of a complex literal is specified by following it with a j or J—for example,

(7 + 3j)

Languages that support a complex type include operations for arithmetic on complex values.

- Decimal

Most larger computers that are designed to support business systems applications have hardware support for decimal data types. Decimal data types store a fixed number of decimal digits, with the implied decimal point at a fixed position in the value.

For example, the number 0.1 (in decimal) can be exactly represented in a decimal type, but not in a floating-point type

Data types for Programming

- **Primitive Data Types:**
  - Characters

The character type is composed of:

• Values: a set of character codes, fixed when the language is defined; the most common of these are ASCII and UNICODE.

• Operations: strongly dependent upon the language; we always find equality, comparison and some way to move from a character to its successor (according to the fixed encoding) and/or to its predecessor.

Values can be stored, expressed and denoted. The representation in store will consist of a single byte (ASCII) or of two bytes (UNICODE).

Data types for Programming

- **Primitive Data Types:**
  - Character String Types

A character string type is one in which the values consist of sequences of characters. Character string constants are used to label output, and the input and output of all kinds of data are often done in terms of strings.

The most common string operations are assignment, catenation, substring reference, comparison, and pattern matching.

A **substring** reference is a reference to a substring of a given string. **Substring references** are discussed in the more general context of arrays, where the substring references are called **slices**.

Python includes strings as a primitive type and has operations for substring reference, catenation, indexing to access individual characters, as well as methods for searching and replacement.

Data types for Programming

Non-Primitive Data Types:

- Composite Types

They are obtained by combining other types using appropriate constructors. The most important and common composite types are:

- Record (or structure), an collection of values in general of different type.
- Array (or vector), a collection of values of the same type.
- Set: subsets of a base type, generally ordinal types.
- Pointer: l-values which permit access to data of another type.
- Recursive types: types defined by recursion, using constants and constructors; particular cases of recursive types are lists, trees, etc.

Data types for Programming

- **Non-Primitive Data Types:**
  - Enumeration Types

An enumeration type is one in which all of the possible values, which are named constants, are provided, or enumerated, in the definition.

Enumeration types provide a way of defining and grouping collections of named constants, which are called enumeration constants.

```
days = [Mon, Tue, Wed, Thu, Fri, Sat, Sun]
colors = [red, blue, green, yellow, black]
```

The enumeration constants are typically implicitly assigned the integer values, 0, 1, . . . but can be explicitly assigned any integer literal in the type's definition.

```
black = 0, white = 1
```

Data types for Programming

- **Non-Primitive Data Types:**
  - Enumeration Types

An enumeration type is one in which all of the possible values, which are named constants, are provided, or enumerated, in the definition.

Enumeration types provide a way of defining and grouping collections of named constants, which are called enumeration constants.

```
days = [Mon, Tue, Wed, Thu, Fri, Sat, Sun]
colors = [red, blue, green, yellow, black]
```

The enumeration constants are typically implicitly assigned the integer values, 0, 1, . . . but can be explicitly assigned any integer literal in the type's definition.

```
black = 0, white = 1
```

Data types for Programming

- **Non-Primitive Data Types:**
  - datetime

The datetime module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the time and calendar modules.

There are two kinds of date and time objects: “**naive**” and “**aware**”.

- An aware object has sufficient knowledge of applicable algorithmic and political time adjustments, such as time zone and daylight saving time information, to locate itself relative to other aware objects.
- A naive object does not contain enough information to unambiguously locate itself relative to other date/time objects..

Data types for Programming

- **Non-Primitive Data Types:**
  - calendar

This module allows you to output calendars like the Unix **cal** program, and provides additional useful functions related to the calendar. By default, these calendars have Monday as the first day of the week, and Sunday as the last (the European convention).

Most of these functions and classes rely on the datetime module which uses an idealized calendar, the current Gregorian calendar extended in both directions.

Use *setfirstweekday()* to set the first day of the week to Sunday or to any other weekday.

Data types for Programming

- **Non-Primitive Data Types:**
  - collections — Container datatypes

This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, dict, list, set, and tuple.

`namedtuple()`	factory function for creating tuple subclasses with named fields
`deque`	list-like container with fast appends and pops on either end
`Counter`	dict subclass for counting hashable objects
`OrderedDict`	dict subclass that remembers the order entries were added
`defaultdict`	dict subclass that calls a factory function to supply missing values
`UserDict`	wrapper around dictionary objects for easier dict subclassing
`UserList`	wrapper around list objects for easier list subclassing
`UserString`	wrapper around string objects for easier string subclassing

A counter tool is provided to support convenient and rapid tallies. For example:

```
>>> # Tally occurrences of words in a list
>>> cnt = Counter()
>>> for word in ['red', 'blue', 'red', 'green', 'blue', 'blue']:
...     cnt[word] += 1
>>> cnt
Counter({'blue': 3, 'red': 2, 'green': 1})
```

Data types for Programming

- **Non-Primitive Data Types:**
  - heapq — Heap queue algorithm

This module provides an implementation of the heap queue algorithm, also known as the priority queue algorithm.

Heaps are binary trees for which every parent node has a value less than or equal to any of its children. This implementation uses arrays for which

heap[k] <= heap[2*k+1] and heap[k] <= heap[2*k+2] for all k,

counting elements from zero.

The interesting property of a heap is that its smallest element is always the root, heap[0].

Data types for Programming

Non-Primitive Data Types:

- Array Types

This module defines an object type which can compactly represent an array of basic values: characters, integers, floating point numbers.

An array is a homogeneous aggregate of data elements in which an individual element is identified by its position in the aggregate, relative to the first element. *The individual data elements of an array are of the same type*.

A two-level syntactic mechanism, where the first part is the aggregate name, and the second part is a possibly dynamic selector consisting of one or more items known as **subscripts** or **indices**.

If all of the **subscripts** in a reference are constants, the selector is **static**; otherwise, it is **dynamic**.

- *A static array* is one in which the subscript ranges are statically bound and storage allocation is static. The advantage of static arrays is efficiency
- *A fixed stack-dynamic array* is one in which the subscript ranges are statically bound, but the allocation is done at declaration elaboration time during execution.
- *A fixed heap-dynamic array* is similar to a fixed stack-dynamic array, in that the subscript ranges and the storage binding are both fixed after storage is allocated.
- *A heap-dynamic array* is one in which the binding of subscript ranges and storage allocation is dynamic and can change any number of times during the array's lifetime. The advantage of heap-dynamic arrays over the others is flexibility.

Data types for Programming

❑ Non-Primitive Data Types:

- Associative Arrays

An associative array is an unordered collection of data elements that are indexed by an equal number of values called keys. So each element of an associative array is in fact a pair of entities, a key and a value.

Python's associative arrays, which are called dictionaries, except the values are all references to objects.

- Record Types

A record is an aggregate of data elements in which the individual elements are identified by names and accessed through offsets from the beginning of the structure.

There is frequently a need in programs to model a collection of data in which the individual elements are not of the same type or size. For example, information about a college student might include name, student number, grade point average, ... A data type for such a collection might use a character string for the name, an integer for the student number, a floating-point for the grade point average, and so forth. Records are designed for this kind of need.

Data types for Programming

Non-Primitive Data Types:

- Tuple Types

A tuple is a data type that is similar to a record, except that the elements are not named. Python includes an immutable tuple type.

If a tuple needs to be changed, it can be converted to an array with the list function. After the change, it can be converted back to a tuple with the tuple function. One use of tuples is when an array must be write protected, such as when it is sent as a parameter to an external function and the user does not want the function to be able to modify the parameter.

```
myTuple = (3, 5.8, 'apple')
```

Tuples can be catenated with the plus (+) operator. They can be deleted with the del statement. There are also other operators and functions that operate on tuples.

Data types for Programming

- **Non-Primitive Data Types:**
  - Pointer and Reference Types

A pointer type is one in which the variables have a range of values that consists of memory addresses and a special value, nil. The value nil is not a valid address and is used to indicate that a pointer cannot currently be used to reference a memory cell.

Pointers are designed for two distinct kinds of uses. First, pointers provide some of the power of indirect addressing, which is frequently used in assembly language programming. Second, pointers provide a way to manage dynamic storage. A pointer can be used to access a location in an area where storage is dynamically allocated called a heap.

Variables that are dynamically allocated from the heap are called heap-dynamic variables. They often do not have identifiers associated with them and thus can be referenced only by pointer or reference type variables. -Variables without names are called anonymous variables. It is in this latter application area of pointers that the most important design issues arise.

Data types for Programming

- **Non-Primitive Data Types:**

  - types — Names for built-in types

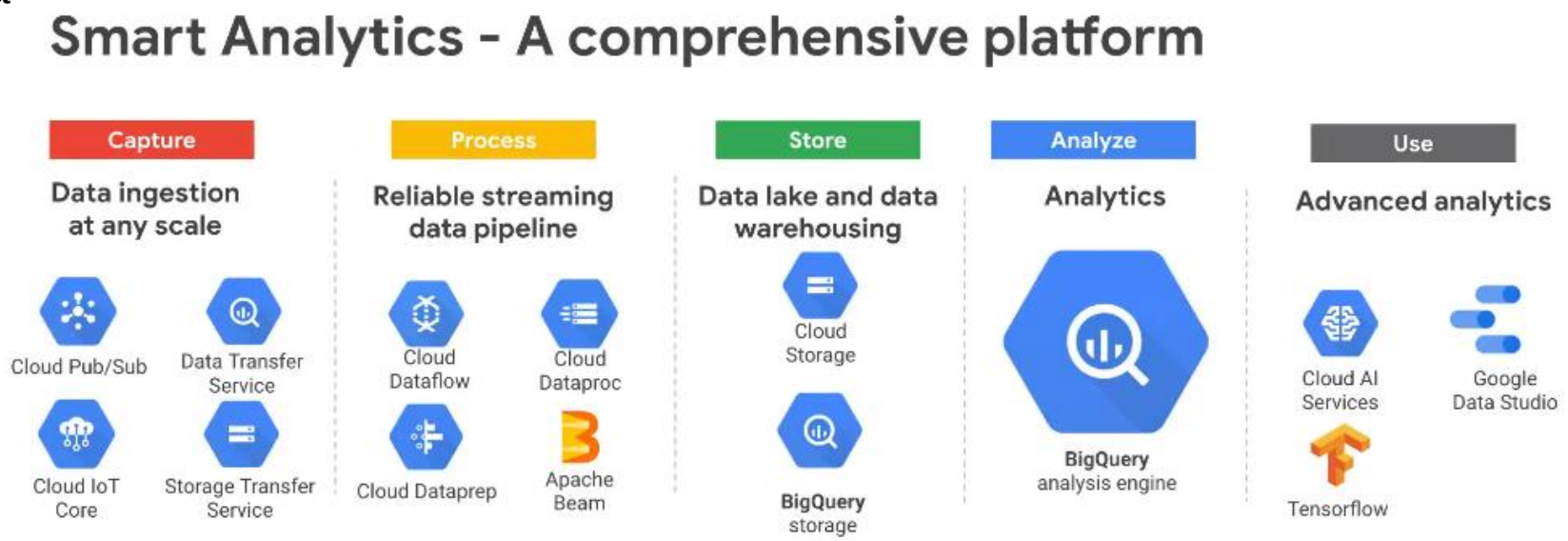
This module defines names for some object types that are used by the standard Python interpreter, but not exposed as builtins like *int* or *str* are. Also, it does not include some of the types that arise transparently during processing such as the *listiterator* type.

Typical use is for *isinstance()* or *issubclass()* checks.

Programming for Data Analysis

- Data Analysis is the technique to collect, transform, and organize data to make future predictions, and make informed data-driven decisions. It also helps to find possible solutions for a business problem. There are six steps for Data Analysis.

1. Ask or Specify Data Requirements
2. Prepare or Collect Data, Read data
3. Clean and Process
4. Analyze, Compute
5. Share, Visualization
6. Act or Report, prediction …



Google Cloud

Programming for Data Analysis

- **Ask or Specify Data Requirements**

What is data?

Types

Sizes

Jobs/business

Trends

Programming for Data Analysis

- **Prepare or Collect Data, Read data**

Data collection (from … sources)

Data classification

Data structures, Data types

Data read/write

Data loading, storage, format …

Programming for Data Analysis

- **Clean and Process**

Analyzing Numerical Data with NumPy, Panda

Array:

Array Indexing, Array Slicing, Array filling, Array sorting

Creating - Updating/Adding - Changing – Removing Data

Data files (read/write/save/delete)

Programming for Data Analysis

- **Analyze and Compute**

Arithmetic Operations

Transforming, Clustering/grouping

Separating, merging, reshaping, splitting

Algorithm, procedure, workflow …

Programming for Data Analysis

- **Share and Visualization**

Data Aggregation

Data sharing

Statistic

DataFrame

Visualization (plot, matrix, graph, figure, ...)

Line, bar, scatter, histogram, pie … graph/chart

Programming for Data Analysis

- **Act or Report, prediction …**

Exploratory Data (relation, summary, trends, warning, forecast …)

Time series (when, where …)

Report (paper, figure,)

Application (business, people, …)

Prediction (Machine learning)