Components of programming

What is a program?

A computer program is a sequence of instructions written to perform a specified task with a computer (Wikipedia)

Compiled versus interpreted programs

Programs usually written in a human writable/readible code → **source code**

... translated into computer-understandible instructions → "to be **compiled"**
- specific to different operating systems : not portable

Some programs are not compiled, but processed by an **interpreter** (compiled program)

Interpreted/scripting languages: **Python, R, MATLAB, Perl, ...**
- portable on different OS

In the following we will write programs for the Python interpreter

Commonly used terms

Arguments	Values that are sent to a program at the time it is run
Code	Noun : A program / line of a program, sometimes called source code Verb : the act of writing a program
Execute	To begin/carry out an operation of a program, synonymous with "run"
Function	Sub-program, can be called repeatedly to perform tasks within a program
Parameters	Values sent to a function when it is called
Parse	Extract particular data elements from a larger block of information
Return	In a Function : sending back a value ; value → variable for function → ...
Run	Execute sequence of commands, or processing a file by a program
Statement	A line of a program or script, which can do value assignments, comparisons, or other operations

Variables

The anatomy of a variable

variable : a name that holds a value

Name - Type - Value

Name : Python - no punctuation, PERL - Name begins with $, no digit in the first character)

Type : Integer, Floating point, string, …

Value : Piece of information, quantitative or qualitative, depending on the type

Scope : Specifies where in a program a variable can be accessed

Variables

Basic variable types

Integer	**Whole numbers**, no fractional component – limit depending on comp. language Specialized integer types for large numbers: - long, - unsigned (twice as a normal integer but not negative)
Floating point	**Rational number**, decimal point can float to any position, precision limit, special case : double precision (2x memory)
Boolean	two values : **True or false**
Strings	**sequence of text characters** : letters, digits, punctuation, … "Within quotation marks" in most languages

```
SequenceName = "Bolinopsis infundibulum"
Primer1 = 'ATGTCTCATTCAAAGCAGG'
DateString = "18-Dec-1965/t13:05"
Location = "Pt. Panic, Oahu, Hawai'i"
```

Often, no extended character sets allowed

Variables as containers for other variables

Arrays and lists

One dimensional arrays : lists, vectors,

Arrays can be composed of different types of values:

```
Morphology = [1, 0, -2, 5.27, 'blue', [4, 2, 4]]
```

Multidimensional arrays : lists of lists, i.e. 2 dimensionsional Matrices

$$A = \begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}$$

```
MyArray [2] = 5
```

Brackets refer to a specific position within the list

Variables as containers for other variables

Dictionary

associative array, hash, map : Container of multiple variables

list ↔ dictionary

sequence of ordered values ↔ collection of names/**keys** point to an associated value

Keys
- Numbers, strings, other types of variables
- must be unique: one key points to one value, multiple keys can have the same value

```
TreeDiam={} ← create an empty dictionary with {}
TreeDiam['Kodiak'] = [68]
TreeDiam['Juneau'] = [85]
```

Values are searched by their keys and not by their position

Variables as containers for other variables

Converting between types

- In some languages, variable types must be specified
- Python assigns (once automatically) the type by the value encountered

- Other languages try to interprete the variable types specifically by background processes.

Different interpretations of values based on their types

add 5 to 123

String : '123' → 1235
Integer : 123 → 128

Variables in action

Operators and **functions** are used in programs to modify or calculate values

Mathematical operators

Addition +, substraction -, multiplication *, division /, power **, equal =, ...

Values of different types can be compatible, but must not
Float + integer but ~~string with integer~~

Sometimes problems:

dividing 2 integers → new integer assigned → 5 / 2 = 2

Variables in action

Comparative and logical operators

Comparison of variables, functons that return a boolean value: True / False

→ Example, to test if one variable is greater than another, decision maker …

→ Are two entities the same or not, equality operator often written as ==, not =

→ In operator : x in A, returns true if value for x is contained in list A,
if A is a list of list, x must be a list, too

→ Other operator, and, or, not, …

order of operators:

→ Follow normal algebraic rules

→ Within those, they are ordered left to right

→ Specified by parenthesis

Variables in action

Functions

- Like little stand alone programs,

- Can be stored in external files, or defined locally in a program

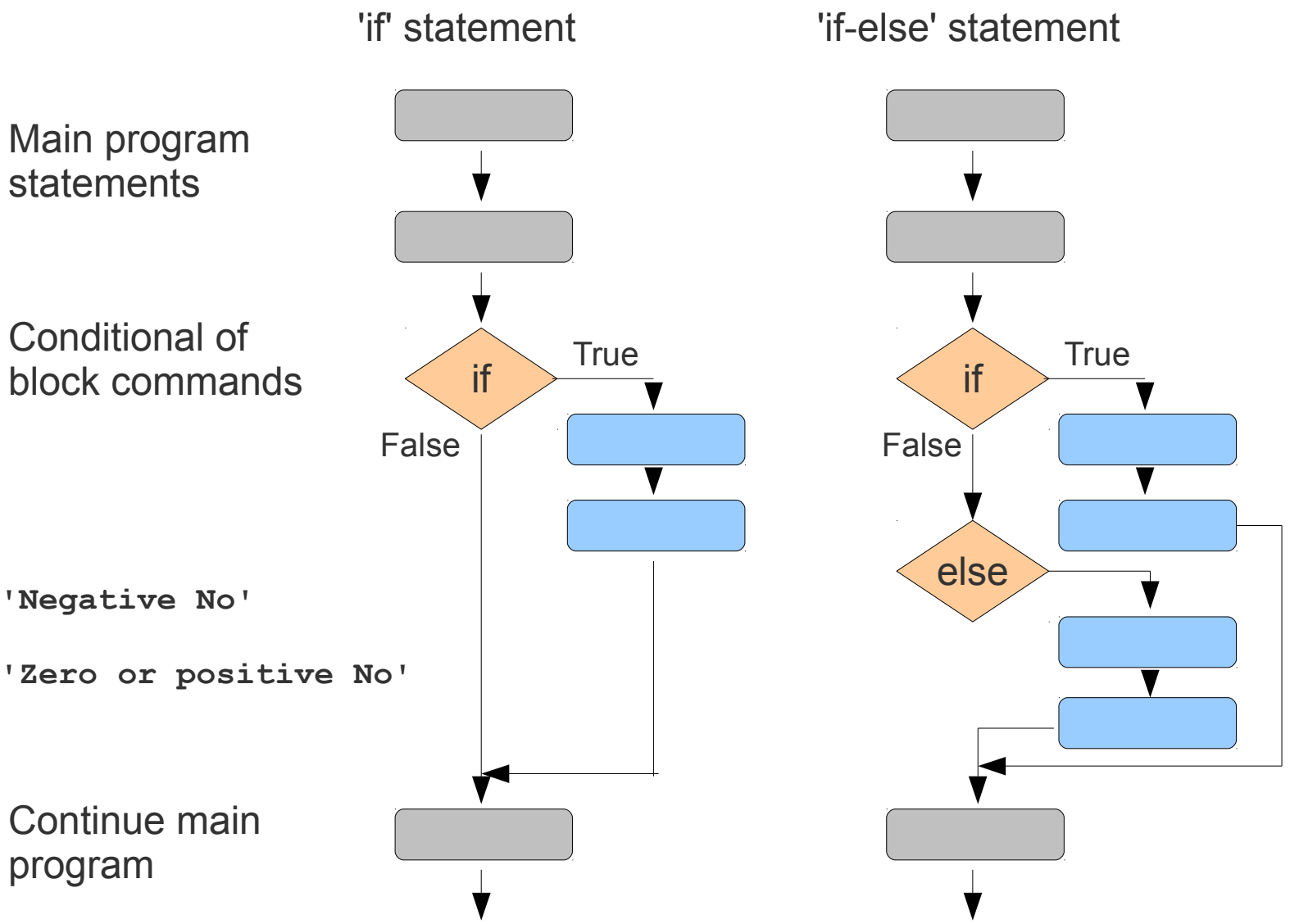
- Functions accept variables, referred to as **parameters**

y = round(2.718) → y = 3

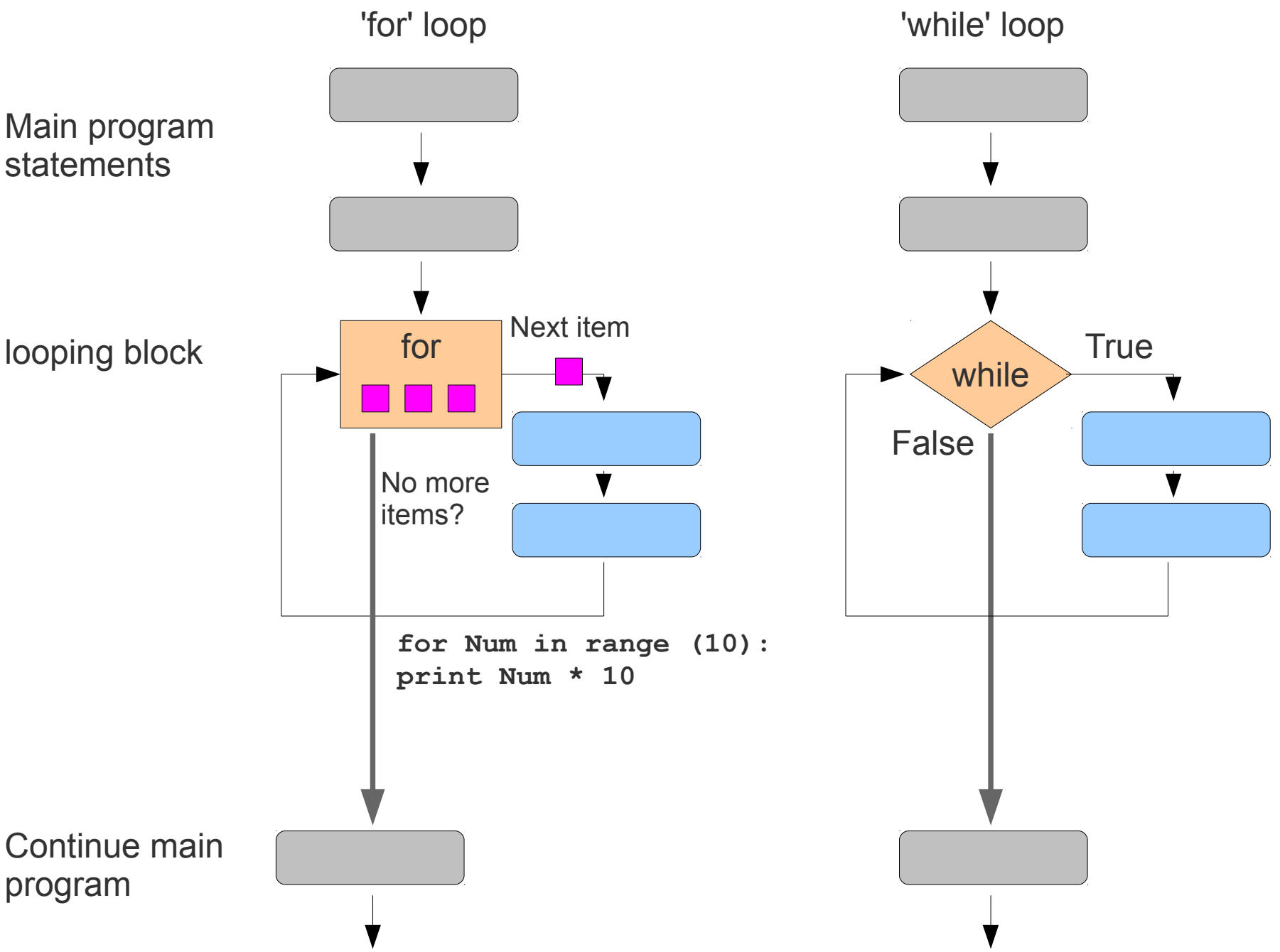
Flow control: Decisions with the 'if' statement

Conditional decision making

```
A=5
if A<0 :
    print 'Negative No'
else :
    print 'Zero or positive No'
```



Flow control: Looping with 'for' and 'while'



Input and output

User interaction

Arguments = program options, specified in the shell

ls **-a**, ls ***.txt**

- Arguments are user input for programs that :
  gather all information → process it autonomously

- Some programs respond to user input while running (for example 'nano')

- Output sent to the screen is usually called 'print'

Input and output

Files

Two levels of operations to access data in a file

- gain access to content
- establish relationships between data in files and variables in the program

Parsing (when reading) ↔ **packaging** of data (when writing a file)

Files can also be used for controlling program behaviour

- store raw input and marching orders
- logbook, logfiles

Libraries and modules

Built-in tools, building blocks : basic functions, simple operators, bundled in modules

Comment statements

Comments are marked with certain characters, depending on the language
#, //, %

Objects

- Sort of "super variable" that contain several other variables within it
- Can contain also functions → then called **methods**

Dot notation:

`MyBike.color` ← Color of your bike
`MyBike.tires` ← properties of your tires

Objects

`MyBike.color` ← Color of your bike
`MyBike.tires` ← properties of your tires

`MyBike.tires.pressure` ← A nested property of your tires

Assignments of Methods (functions):

`MyBike.steer(-4)` ← Steer 4 degree to the left
`MyBike.color('red')` ← You can often set and read with the same notation
`MyBike.pedals.pedal(100)` ← with the pedals, pedal at a speed of 100

Sometimes more convenient to use objects instead of a separate function,

Example :

```
MyString='abc'
Print uppercase(MyString)
```

or :

```
print MyString.upper()
```