Components of Programming

Different types of programs

- Compiled programs (source code):
  - self-contained, computer understandable program
- Interpreted programs:
  - Processed by an interpreter each time it is run
  - written in a “scripting language” Python, R, MATLAB, Perl, etc.
- Advantages/disadvantages?
- portability, edit-ability, speed

Terminology

TABLE 7.1 Glossary of program-related terms

Term	Definfition
Arguments	Values that are sent to a program at the time it is run
Code	*Noun*: A program or line of a program, sometimes called source code *Verb*: The act of writing a program
Execute	To begin and carry out the operation of a program; synonymous with *run*
Function	A subprogram that can be called repeatedly to perform the same task within a program
Parameters	Values that are sent to a function when it is called
Parse	To extract particular data elements from a larger block of text
Return	In a function, the act of sending back a value; the value can be assigned to a variable by referring to the function name (e.g., in `y = cos(x)`, the function `cos` calculates and returns the value of the cosine of `x`, which is assigned to `y`)
Run	To execute the sequence of commands in a program; can also refer to the processing of a file by a program that finds instructions within it
Statement	A line of a program or script, which can assign a value, do a comparison, or perform other operations

Variables

- name, type, value

TABLE 7.2 Commonly used variable types

Type	Example
Integer	`98`
Float	`98.6`
Boolean	`False`
String	`'Bargmannia elongata'`

Variables ctd.

- name, type, value

TABLE 7.2 Commonly used variable types

Type	Example	
Integer	`98`	-+2,147,483,648 long unsigned (+)
Float	`98.6`	
Boolean	`False`	
String	`'Bargmannia elongata'`	

Variables ctd.

- name, type, value

TABLE 7.2 Commonly used variable types

Type	Example
Integer	`98`
Float	`98.6`
Boolean	`False`
String	`'Bargmannia elongata'`

4E-15
double

Variables ctd.

- name, type, value

TABLE 7.2 Commonly used variable types

Type	Example
Integer	`98`
Float	`98.6`
Boolean	`False`=0, True=1
String	`'Bargmannia elongata'`

Variables ctd.

- name, type, value

TABLE 7.2 Commonly used variable types

Type	Example
Integer	`98`
Float	`98.6`
Boolean	`False`
String	`'Bargmannia elongata'`

```
SequenceName='Bolinopsis infundibulum'
DateString='18-Dec-1865\t13:05'
```

arrays

- 1-dimensional (lists, vectors)
- variables containing other variables

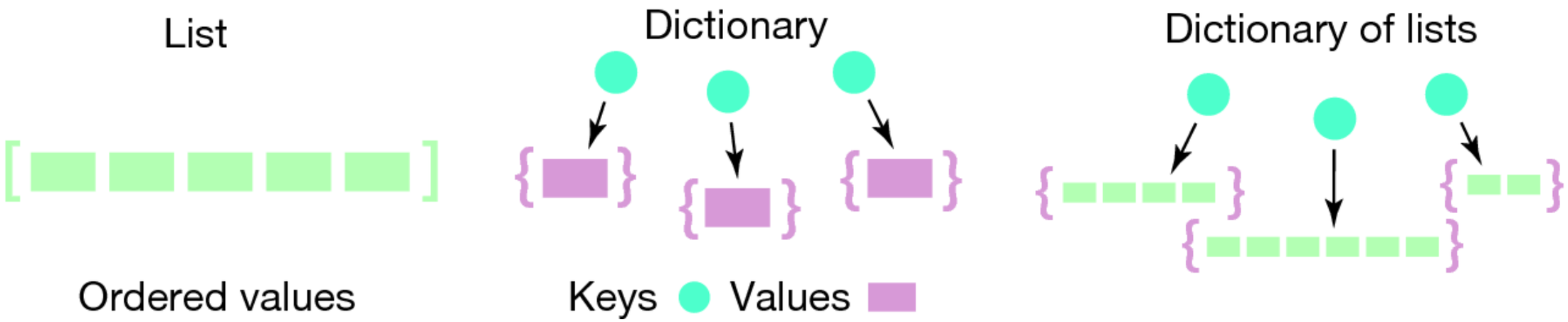
```
alist=[]
Morphology=[1, 0, -2, 5.27, 'blue'. [4,2,4]]
Morphology[2]=?
```

- multidimensional

Species1, Species2, Species3

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

arrays ctd.



- dictionaries, [key]=value
  - each [key] must be unique

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

variable types and operators

- type string

```
TreeDiam="123"
TreeDiam+5=?
```

- types and operators

```
x=5
x/2=?
```

- = vs. ==

```
x=5
x==5
x==2
```

- in

```
2 in x?
2 in TreeDiam?
"2" in TreeDiam?
```

TABLE 7.3 Common operators and their symbols

Operator*	Common symbols
Mathematical	
Addition	`+`
Subtraction	`-`
Multiplication	`*`
Division	`/`
Power	`**`
Modulo (remainder after division)	`%`
Truncated division (result without remainder)	`//`
Comparative	
Equal to	`==`
Not equal to	`!=, <>, ~=`
Greater than	`>`
Less than	`<`
Greater or equal	`>=`
Less or equal	`<=`
Logical	
And	`and, &, &&`
Or	`or, |, ||`
Not	`not, !, ~`

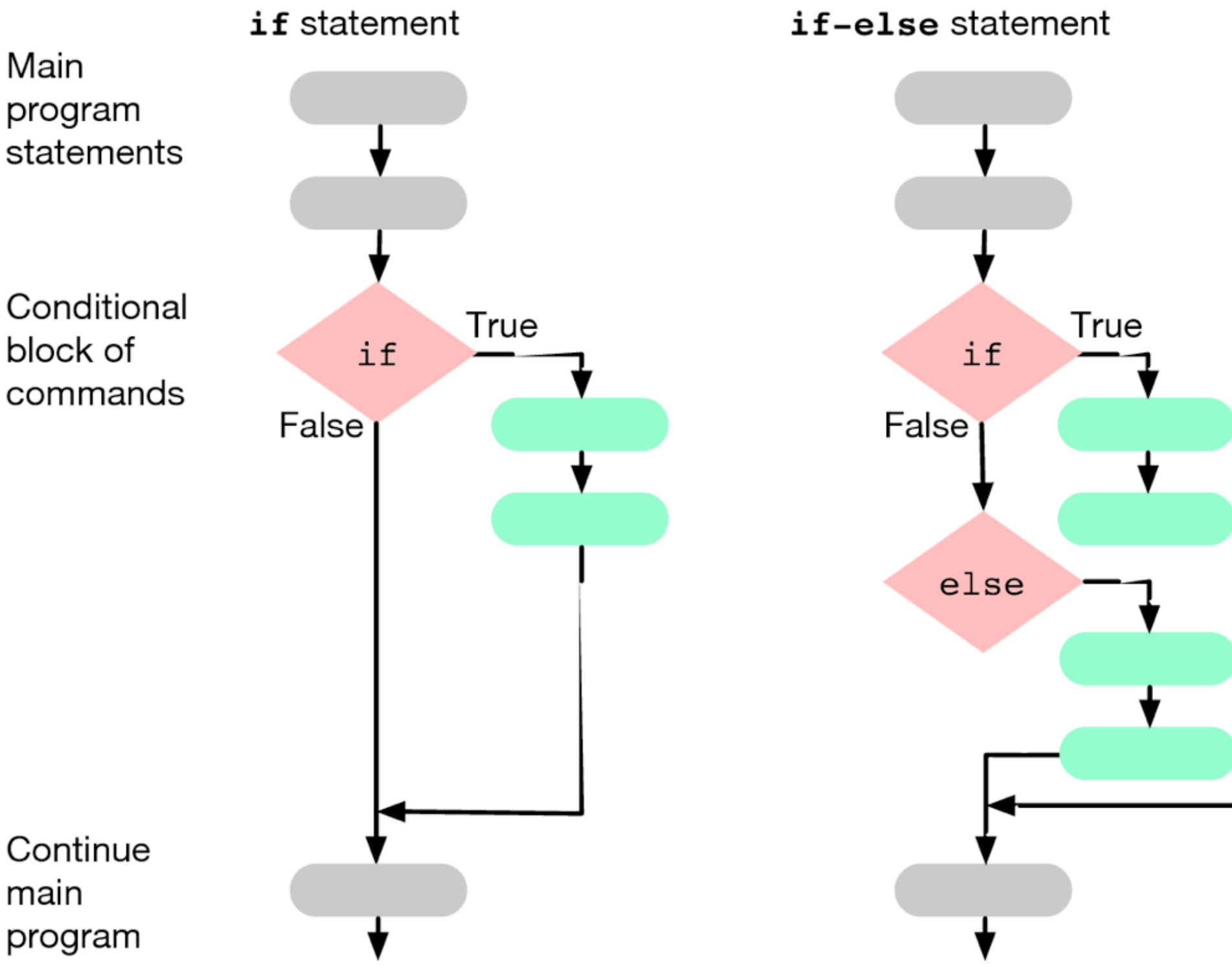
*Not all languages support all operator symbols, and in some cases the name of the operator is used instead of a symbol.

functions

```
y = round(2.718)
x = ‘somestring’.split(‘e’)
```

- take input/parameters

flow control

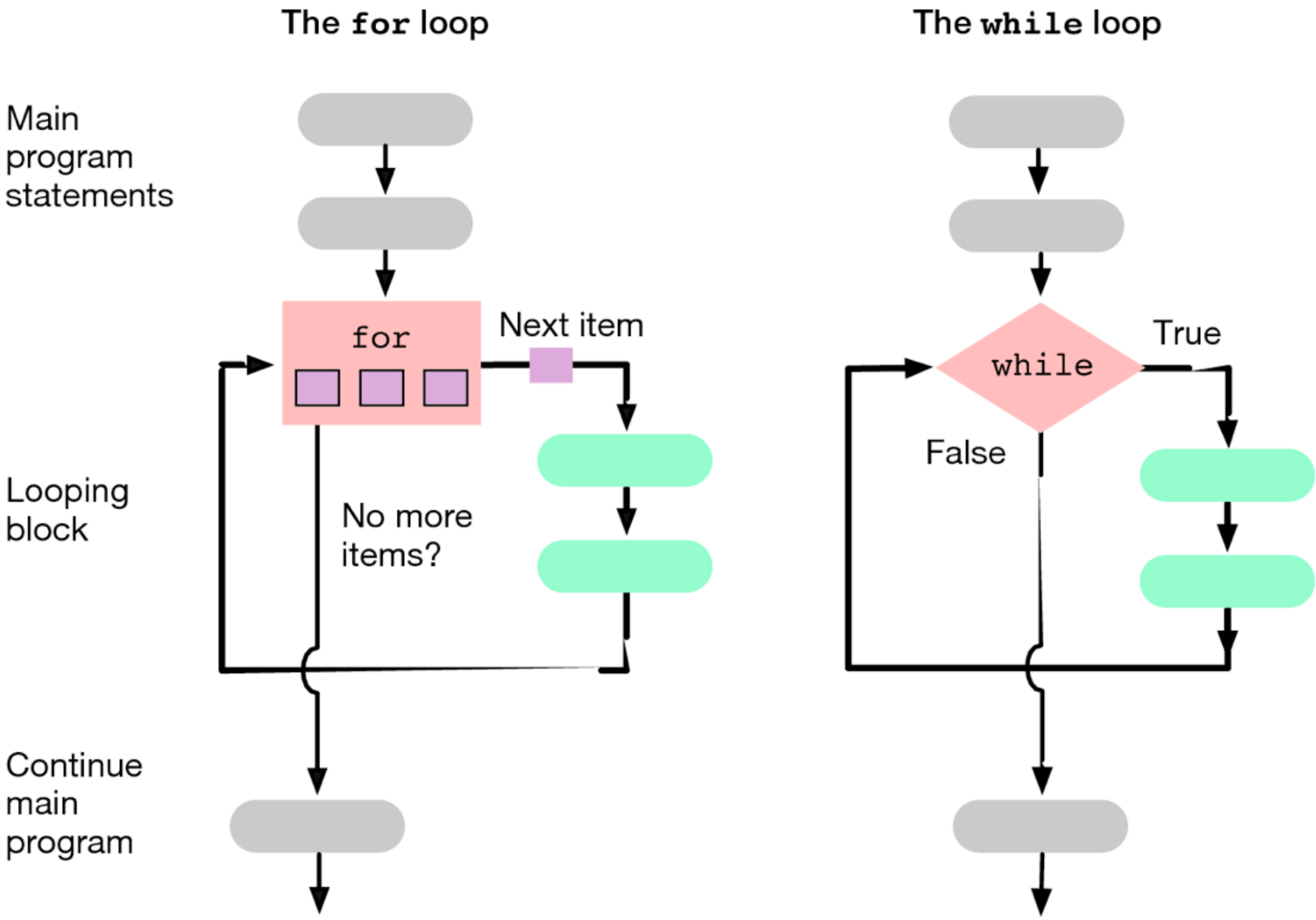


flow control

- if statements

```
A = 5
if A < 0
   print ‘Negative number’
else:
   print ‘Zero or positive number’
```

loops


The for loop
The while loop
Main program statements
for
Next item
No more items?
Looping block
Continue main program
while
True
False

Lists and Dictionaries

```
TreeStat={} # create an empty dictionary with {}
TreeStat['Kodiak']=[68, 57.8, -152.5]
TreeStat['Juneau']=[85, 58.3, -134.5]
TreeStat['Barrow']=[133, 71.3, -156.6]
```

- strings as lists

```
Line='Kodiak,68,57.8,-152.5'
```

Files, Libraries, and Objects

- arguments
- what are files used for?
- parsing
- modules

```
import sys, os, numpy
```

- objects

```
MyBike.color
MyBike.tires
MyBike.tires.pressure

MyString='abc'
print uppercaser(MyString)
print MyString.upper()
```

Problem-solving approach/aka how to write a script

1. Name and write down the basic problem you are trying to solve. Describe your input file format and desired output file format.
2. Write down the possible variable types, functions, and flow direction approaches that might apply to the goals of the script.
3. Write out a conceptual plan for the script, identify the major steps.
4. Translate the steps into code using the tools identified in step 2.
5. Run and debug your script.
6. Step back, and evaluate the approach, identify places for improvement, streamlining, etc.