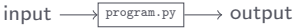
Input and Output

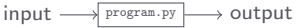
Outline

Input and Output

Input and Output

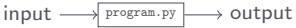


Input and Output



Input types:

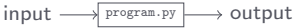
Input and Output



Input types:

- Command-line input

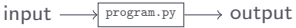
Input and Output



Input types:

- Command-line input
- Standard input

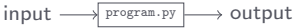
Input and Output



Input types:

- Command-line input
- Standard input
- File input

Input and Output



Input types:

- Command-line input
- Standard input
- File input

Output types:

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output

Input and Output

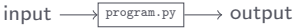
Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Graphical output

Input and Output



Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Graphical output
- Audio output

Input and Output

Input types:

- Command-line input
- Standard input
- File input

Output types:

- Standard output
- Graphical output
- Audio output
- File output

Standard Output Revisited

Standard Output Revisited

stdio	
`writeln(x = "")`	writes `x` followed by newline to standard output
`write(x = "")`	writes `x` to standard output
`writef(fmt, *args)`	writes each element of `args` to standard output according to the format specified by the string `fmt`

Standard Output Revisited

stdio

`writeln(x = "")`	writes `x` followed by newline to standard output
`write(x = "")`	writes `x` to standard output
`writef(fmt, *args)`	writes each element of `args` to standard output according to the format specified by the string `fmt`

Example

```
stdio.writef("pi is approximately %.2f.\n", math.pi)
stdio.writef("The %dth decimal digit of %.10f is %d.\n", 5, math.pi, 9)
```

Standard Output Revisited

stdio

`writeln(x = "")`	writes `x` followed by newline to standard output
`write(x = "")`	writes `x` to standard output
`writef(fmt, *args)`	writes each element of `args` to standard output according to the format specified by the string `fmt`

Example

```
stdio.writef("pi is approximately %.2f.\n", math.pi)
stdio.writef("The %dth decimal digit of %.10f is %d.\n", 5, math.pi, 9)
```

```
pi is approximately 3.14.
The 5th decimal digit of 3.1415926536 is 9.
```

Standard Output Revisited

Standard Output Revisited

Program: `randomseq.py`

Standard Output Revisited

Program: `randomseq.py`

- Command-line input: *n* (int), *lo* (float), and *hi* (float)

Standard Output Revisited

Program: `randomseq.py`

- Command-line input: n (int), lo (float), and hi (float)
- Standard output: n random floats from the interval $[lo, hi)$, each up to 2 decimal places

Standard Output Revisited

Program: `randomseq.py`

- Command-line input: n (int), lo (float), and hi (float)
- Standard output: n random floats from the interval $[lo, hi)$, each up to 2 decimal places

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Output Revisited

Program: `randomseq.py`

- Command-line input: n (int), lo (float), and hi (float)
- Standard output: n random floats from the interval $[lo, hi)$, each up to 2 decimal places

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 10 100 200
```

Standard Output Revisited

Program: `randomseq.py`

- Command-line input: n (int), lo (float), and hi (float)
- Standard output: n random floats from the interval $[lo, hi)$, each up to 2 decimal places

```
>_ ~/workspace/ipp/programs

$ python3 randomseq.py 10 100 200
186.69
102.34
176.05
182.78
161.95
169.34
155.65
154.96
194.41
103.91
$ _
```

Standard Output Revisited

Standard Output Revisited

randomseq.py

```
import stdio
import stdrandom
import sys

n = int(sys.argv[1])
lo = float(sys.argv[2])
hi = float(sys.argv[3])
for i in range(n):
    r = stdrandom.uniformFloat(lo, hi)
    stdio.writef("%.2f\n", r)
```

Standard Input

Standard Input

Standard input is input entered interactively on the terminal

Standard Input

Standard input is input entered interactively on the terminal

The end of standard input stream is signalled by the end-of-file (EOF) character (`<ctrl-d>`)

Standard Input

Standard input is input entered interactively on the terminal

The end of standard input stream is signalled by the end-of-file (EOF) character (`<ctrl-d>`)

stdio

`isEmpty()`	returns `True` if standard input is empty, and `False` otherwise
`readInt()`	returns a token from standard input as an integer
`readAllInts()`	returns the remaining tokens from standard input as a list of integers
`readFloat()`	returns a token from standard input as a float
`readAllFloats()`	returns the remaining tokens from standard input as a list of floats
`readString()`	returns a token from standard input as a string
`readAllStrings()`	returns the remaining tokens from standard input as a list of strings
`readAll()`	returns the remaining tokens from standard input as a string

Standard Input

Standard Input

Program: `twentyquestions.py`

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs
$ python3 twentyquestions.py
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
What is your guess? _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
What is your guess? 501694
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
What is your guess? 501694
Too high
What is your guess? _
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
What is your guess? 501694
Too high
What is your guess? 501686
```

Standard Input

Program: `twentyquestions.py`

- Standard input: user guesses
- Standard output: "Too low", "Too high", or "You win!"

```
>_ ~/workspace/ipp/programs

$ python3 twentyquestions.py
I am thinking of a secret number between 1 and 1000000
What is your guess? 500000
Too low
What is your guess? 750000
Too high
What is your guess? 625000
Too high
...
What is your guess? 501694
Too high
What is your guess? 501686
You win!
$ _
```

Standard Input

Standard Input

twentyquestions.py

```
import stdio
import stdrandom

RANGE = 1000000
secret = stdrandom.uniformInt(1, RANGE + 1)
stdio.writef("I am thinking of a secret number between 1 and %d\n", RANGE)
guess = 0
while guess != secret:
    stdio.write("What is your guess? ")
    guess = stdio.readInt()
    if guess < secret:
        stdio.writeln("Too low")
    elif guess > secret:
        stdio.writeln("Too high")
    else:
        stdio.writeln("You win!")
```

Standard Input

Standard Input

Program: `average.py`

Standard Input

Program: `average.py`

- Standard input: a sequence of floats

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs
$ python3 average.py
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs
$ python3 average.py
-
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs

$ python3 average.py
1.0 5.0 6.0
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs

$ python3 average.py
1.0 5.0 6.0
-
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs

$ python3 average.py
1.0 5.0 6.0
3.0 7.0 32.0
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs

$ python3 average.py
1.0 5.0 6.0
3.0 7.0 32.0
-
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs

$ python3 average.py
1.0 5.0 6.0
3.0 7.0 32.0
<ctrl-d>
```

Standard Input

Program: `average.py`

- Standard input: a sequence of floats
- Standard output: their average value

```
>_ ~/workspace/ipp/programs
$ python3 average.py
1.0 5.0 6.0
3.0 7.0 32.0
<ctrl-d>
Average is 10.5
$ _
```

Standard Input

Standard Input

average.py

```
import stdio

total = 0.0
count = 0
while not stdio.isEmpty():
    x = stdio.readFloat()
    total += x
    count += 1
average = total / count
stdio.writeln("Average is " + str(average))
```

Standard Input

Standard Input

Program: `rangefilter.py`

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range $[lo, hi]$

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs
$ python3 rangefilter.py 100 400
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs
$ python3 rangefilter.py 100 400
-
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs

$ python3 rangefilter.py 100 400
358 1330 55 165 689 1014 3066 387 575 843 203 48 292 877 65 998
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs

$ python3 rangefilter.py 100 400
358 1330 55 165 689 1014 3066 387 575 843 203 48 292 877 65 998
358 165 387 203 292 _
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs

$ python3 rangefilter.py 100 400
358 1330 55 165 689 1014 3066 387 575 843 203 48 292 877 65 998
358 165 387 203 292 <ctrl-d>
```

Standard Input

Program: `rangefilter.py`

- Command-line input: *lo* (int) and *hi* (int)
- Standard input: a sequence of integers
- Standard output: those integers that are in the range [*lo*, *hi*]

```
>_ ~/workspace/ipp/programs

$ python3 rangefilter.py 100 400
358 1330 55 165 689 1014 3066 387 575 843 203 48 292 877 65 998
358 165 387 203 292 <ctrl-d>
$ _
```

Standard Input

Standard Input

rangefilter.py

```
import stdio
import sys

lo = int(sys.argv[1])
hi = int(sys.argv[2])
while not stdio.isEmpty():
    x = stdio.readInt()
    if x >= lo and x <= hi:
        stdio.write(str(x) + " ")
stdio.writeln()
```

Standard Input

Standard Input

Output redirection operator (>)

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs
$ python3 average.py < data.txt
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs
$ python3 average.py < data.txt
Average is 149.1812199999999
$ _
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs
$ python3 average.py < data.txt
Average is 149.1812199999999
$ _
```

Piping operator (|)

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs

$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs

$ python3 average.py < data.txt
Average is 149.1812199999999
$ _
```

Piping operator (|)

```
>_ ~/workspace/ipp/programs

$ _
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs
$ python3 average.py < data.txt
Average is 149.1812199999999
$ _
```

Piping operator (|)

```
>_ ~/workspace/ipp/programs
$ python3 randomseq.py 1000 100.0 200.0 | python3 average.py
```

Standard Input

Output redirection operator (>)

```
>_ ~/workspace/ipp/programs

$ python3 randomseq.py 1000 100.0 200.0 > data.txt
$ _
```

Input redirection operator (<)

```
>_ ~/workspace/ipp/programs

$ python3 average.py < data.txt
Average is 149.1812199999999
$ _
```

Piping operator (|)

```
>_ ~/workspace/ipp/programs

$ python3 randomseq.py 1000 100.0 200.0 | python3 average.py
Average is 149.17643999999999
$ _
```

Standard Draw

Standard Draw

The `stddraw` library provides an abstraction for producing drawings as output

Standard Draw

The `stddraw` library provides an abstraction for producing drawings as output

stddraw	
`WHITE`	represents white
`BLACK`	represents black
`BLUE`	represents blue
`setCanvasSize(w = 512, h = 512)`	sets the width and height of the canvas to `w` and `h` pixels
`setXscale(min = 0.0, max = 1.0)`	sets the *x*-scale of canvas to the interval `[min, max]`
`setYscale(min = 0.0, max = 1.0)`	sets the *y*-scale of canvas to the interval `[min, max]`
`setPenRadius(r = 0.005)`	sets the pen radius to `r`
`setPenColor(c = BLACK)`	sets the pen color to `c`
`point(x, y)`	draws on the canvas a point at `(x, y)`
`line(x0, y0, x1, y1)`	draws on the canvas a line from `(x0, y0)` to `(x1, y1)`
`filledCircle(x, y, r)`	draws on the canvas a filled circle of radius `r` centered at `(x, y)`
`filledSquare(x, y, r)`	draws on the canvas a filled square of side length `2r` centered at `(x, y)`
`text(x, y, s)`	draw on canvas the string `s` centered at `(x, y)`
`clear(c = WHITE)`	clears the canvas to color `c`
`show(msec = float("inf"))`	shows the canvas and waits for `msec` milliseconds

Standard Draw

Standard Draw

Program: `plotfilter.py`

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

```
>_ ~/workspace/ipp/programs
$ cat ../data/usa.txt
```

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

```
>_ ~/workspace/ipp/programs

$ cat ../data/usa.txt
669905.0 245552.0 1244962.0 490000.0
1097038.8890 245552.7780
1103961.1110 247133.3330
...
692230.5560 490000.0000
$ _
```

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

```
>_ ~/workspace/ipp/programs

$ cat ../data/usa.txt
669905.0 245552.0 1244962.0 490000.0
1097038.8890 245552.7780
1103961.1110 247133.3330
...
692230.5560 490000.0000
$ python3 plotfilter.py < ../data/usa.txt
```

Standard Draw

Program: `plotfilter.py`

- Standard input: x and y scales and (x, y) points
- Standard draw output: a plot of the points

```
>_ ~/workspace/ipp/programs

$ cat ../data/usa.txt
669905.0 245552.0 1244962.0 490000.0
1097038.8890 245552.7780
1103961.1110 247133.3330
...
692230.5560 490000.0000
$ python3 plotfilter.py < ../data/usa.txt
$ _
```

Standard Draw

Standard Draw

plotfilter.py

```
import stddraw
import stdio

x0 = stdio.readFloat()
y0 = stdio.readFloat()
x1 = stdio.readFloat()
y1 = stdio.readFloat()
stddraw.setXscale(x0, x1)
stddraw.setYscale(y0, y1)
stddraw.setPenRadius(0.0)
while not stdio.isEmpty():
    x = stdio.readFloat()
    y = stdio.readFloat()
    stddraw.point(x, y)
stddraw.show()
```

Standard Draw

Standard Draw

Program: `bouncingball.py`

Standard Draw

Program: `bouncingball.py`

- Standard draw output: a bouncing ball

Standard Draw

Program: `bouncingball.py`

- Standard draw output: a bouncing ball

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Draw

Program: `bouncingball.py`

- Standard draw output: a bouncing ball

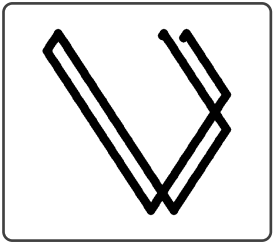
```
>_ ~/workspace/ipp/programs
$ python3 bouncingball.py
```

Standard Draw

Program: `bouncingball.py`

- Standard draw output: a bouncing ball

```
>_ ~/workspace/ipp/programs
$ python3 bouncingball.py
$ _
```



Standard Draw

Standard Draw

bouncingball.py

```
import stddraw

RADIUS = 0.05
DT = 1.0
PAUSE = 20
stddraw.setXscale(-1.0, 1.0)
stddraw.setYscale(-1.0, 1.0)
rx = 0.480
ry = 0.860
vx = 0.015
vy = 0.023
while True:
    if abs(rx + vx * DT) + RADIUS > 1.0:
        vx = -vx
    if abs(ry + vy * DT) + RADIUS > 1.0:
        vy = -vy
    rx += vx * DT
    ry += vy * DT
    stddraw.clear(stddraw.WHITE)
    stddraw.filledCircle(rx, ry, RADIUS)
    stddraw.show(PAUSE)
```

Standard Audio

Standard Audio

To obtain a digital sound, we sample a continuous sound signal (represented by a sine curve) at regular intervals — a widely used sampling rate is 44,100 samples per second

Standard Audio

To obtain a digital sound, we sample a continuous sound signal (represented by a sine curve) at regular intervals — a widely used sampling rate is 44,100 samples per second

A digital sound is thus a list of real numbers between -1 and $+1$

Standard Audio

To obtain a digital sound, we sample a continuous sound signal (represented by a sine curve) at regular intervals — a widely used sampling rate is 44,100 samples per second

A digital sound is thus a list of real numbers between -1 and $+1$

The `stdaudio` library provides an abstraction for playing, manipulating, and synthesizing digital sounds

Standard Audio

To obtain a digital sound, we sample a continuous sound signal (represented by a sine curve) at regular intervals — a widely used sampling rate is 44,100 samples per second

A digital sound is thus a list of real numbers between -1 and $+1$

The `stdaudio` library provides an abstraction for playing, manipulating, and synthesizing digital sounds

`stdaudio`

`wait()`	waits for the currently playing sound to finish
`playSamples(a)`	plays all sound samples in the list `a`
`playFile(file)`	plays all sound samples in the file whose name is `file.wav`

Standard Audio

Standard Audio

Program: `playthattune.py`

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

```
>_ ~/workspace/ipp/programs
$ _
```

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

```
>_ ~/workspace/ipp/programs
$ cat ../data/elise.txt
```

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

```
>_ ~/workspace/ipp/programs

$ cat ../data/elise.txt
7 .125
6 .125
7 .125
...
0 .25
$ _
```

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

```
>_ ~/workspace/ipp/programs

$ cat ../data/elise.txt
7 .125
6 .125
7 .125
...
0 .25
$ python3 playthattune.py < ../data/elise.txt
```

Standard Audio

Program: `playthattune.py`

- Standard input: sound samples, each characterized by a pitch and a duration
- Standard audio output: the sound

```
>_ ~/workspace/ipp/programs

$ cat ../data/elise.txt
7 .125
6 .125
7 .125
...
0 .25
$ python3 playthattune.py < ../data/elise.txt
$ _
```

Standard Audio

Standard Audio

playthattune.py

```
import math
import stdarray
import stdaudio
import stdio

SPS = 44100
NOTES_ON_SCALE = 12
CONCERT_A = 440.0
while not stdio.isEmpty():
    pitch = stdio.readInt()
    duration = stdio.readFloat()
    hz = CONCERT_A * math.pow(2, pitch / NOTES_ON_SCALE)
    n = int(SPS * duration)
    note = stdarray.create1D(n + 1, 0.0)
    for i in range(n + 1):
        note[i] = math.sin(2 * math.pi * i * hz / SPS)
    stdaudio.playSamples(note)
stdaudio.wait()
```