Defining Functions

Outline

Defining Functions

Defining Functions

Function definition

```
def <name>(<parameter name>, <parameter name>, ...):
    <statement>
    <statement>
    ...
```

Defining Functions

Function definition

```
def <name>(<parameter name>, <parameter name>, ...):
    <statement>
    <statement>
    ...
```

Return statement

```
return [<expression>]
```

Defining Functions

Function definition

```
def <name>(<parameter name>, <parameter name>, ...):
    <statement>
    <statement>
    ...
```

Return statement

```
return [<expression>]
```

Example

```
def is_prime(n):
    if n < 2:
        return False
    i = 2
    while i <= n // i:
        if n % i == 0:
            return False
        i += 1
    return True
```

Defining Functions

Defining Functions

The scope of a function's local and parameter variables is limited to that function

Defining Functions

The scope of a function's local and parameter variables is limited to that function

The scope of a variable defined in global code — known as a global variable — is limited to the `.py` file containing that variable

Defining Functions

Defining Functions

A function may designate an argument to be optional by specifying a default value for that argument

Defining Functions

A function may designate an argument to be optional by specifying a default value for that argument

Example (computing $H_{n,r} = 1 + 1/2^r + 1/3^r + \cdots + 1/n^r$)

```
def harmonic(n, r = 1):
    total = 0.0
    for i in range(1, n + 1):
        total += 1 / (i ** r)
    return total
```

Defining Functions

Defining Functions

If a function parameter refers to a mutable object, changing that object's value within the function also changes the object's value in the calling code

Defining Functions

If a function parameter refers to a mutable object, changing that object's value within the function also changes the object's value in the calling code

Example

```
def exchange(a, i, j):
    temp = a[i]
    a[i] = a[j]
    a[j] = temp

a = [1, 2, 3, 4, 5]
exchange(a, 1, 3)
stdio.writeln(a)
```

Defining Functions

If a function parameter refers to a mutable object, changing that object's value within the function also changes the object's value in the calling code

Example

```
def exchange(a, i, j):
    temp = a[i]
    a[i] = a[j]
    a[j] = temp

a = [1, 2, 3, 4, 5]
exchange(a, 1, 3)
stdio.writeln(a)
```

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

Defining Functions

Defining Functions

Program: `harmonicredux.py`

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
```

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
2.9289682539682538
$ _
```

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
2.9289682539682538
$ python3 harmonicredux.py 1000
```

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
2.9289682539682538
$ python3 harmonicredux.py 1000
7.485470860550343
$ _
```

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
2.9289682539682538
$ python3 harmonicredux.py 1000
7.485470860550343
$ python3 harmonicredux.py 10000
```

Defining Functions

Program: `harmonicredux.py`

- Command-line input: n (int)
- Standard output: the nth harmonic number $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

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

```
$ python3 harmonicredux.py 10
2.9289682539682538
$ python3 harmonicredux.py 1000
7.485470860550343
$ python3 harmonicredux.py 10000
9.787606036044348
$ _
```

Defining Functions

Defining Functions

harmonicredux.py

```
import stdio
import sys

def main():
    n = int(sys.argv[1])
    stdio.writeln(_harmonic(n))

def _harmonic(n):
    total = 0.0
    for i in range(1, n + 1):
        total += 1 / i
    return total

if __name__ == '__main__':
    main()
```

Defining Functions

harmonicredux.py

```
1  import stdio
2  import sys
3
4  def main():
5      n = int(sys.argv[1])
6      stdio.writeln(_harmonic(n))
7
8  def _harmonic(n):
9      total = 0.0
10     for i in range(1, n + 1):
11         total += 1 / i
12     return total
13
14 if __name__ == '__main__':
15     main()
```

When a program is imported as a library, the program's `__name__` attribute is not set to `'__main__'`

>_ ~/workspace/ipp/programs

```
>>> _
```

Defining Functions

harmonicredux.py

```
1  import stdio
2  import sys
3
4  def main():
5      n = int(sys.argv[1])
6      stdio.writeln(_harmonic(n))
7
8  def _harmonic(n):
9      total = 0.0
10     for i in range(1, n + 1):
11         total += 1 / i
12     return total
13
14 if __name__ == '__main__':
15     main()
```

When a program is imported as a library, the program's `__name__` attribute is not set to `'__main__'`

>_ ~/workspace/ipp/programs

```
>>> import harmonicredux
```

Defining Functions

harmonicredux.py

```
1  import stdio
2  import sys
3
4  def main():
5      n = int(sys.argv[1])
6      stdio.writeln(_harmonic(n))
7
8  def _harmonic(n):
9      total = 0.0
10     for i in range(1, n + 1):
11         total += 1 / i
12     return total
13
14 if __name__ == '__main__':
15     main()
```

When a program is imported as a library, the program's `__name__` attribute is not set to `'__main__'`

>_ ~/workspace/ipp/programs

```
>>> import harmonicredux
>>> _
```

Defining Functions

harmonicredux.py

```
1  import stdio
2  import sys
3
4  def main():
5      n = int(sys.argv[1])
6      stdio.writeln(_harmonic(n))
7
8  def _harmonic(n):
9      total = 0.0
10     for i in range(1, n + 1):
11         total += 1 / i
12     return total
13
14 if __name__ == '__main__':
15     main()
```

When a program is imported as a library, the program's `__name__` attribute is not set to `'__main__'`

>_ ~/workspace/ipp/programs

```
>>> import harmonicredux
>>> harmonicredux._harmonic(10)
```

Defining Functions

harmonicredux.py

```
1  import stdio
2  import sys
3
4  def main():
5      n = int(sys.argv[1])
6      stdio.writeln(_harmonic(n))
7
8  def _harmonic(n):
9      total = 0.0
10     for i in range(1, n + 1):
11         total += 1 / i
12     return total
13
14 if __name__ == '__main__':
15     main()
```

When a program is imported as a library, the program's `__name__` attribute is not set to `'__main__'`

>_ ~/workspace/ipp/programs

```
>>> import harmonicredux
>>> harmonicredux._harmonic(10)
2.9289682539682538
>>> _
```

Defining Functions

Defining Functions

Program: `couponcollectorredux.py`

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

```
>_ ~/workspace/ipp/programs
$ python3 couponcollectorredux.py 1000
```

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

```
$ python3 couponcollectorredux.py 1000
6276
$ _
```

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

```
$ python3 couponcollectorredux.py 1000
6276
$ python3 couponcollectorredux.py 1000
```

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

$ python3 couponcollectorredux.py 1000
6276
$ python3 couponcollectorredux.py 1000
7038
$ _
```

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

```
$ python3 couponcollectorredux.py 1000
6276
$ python3 couponcollectorredux.py 1000
7038
$ python3 couponcollectorredux.py 1000000
```

Defining Functions

Program: `couponcollectorredux.py`

- Command-line input: n (int)
- Standard output: number of coupons one must collect before obtaining one of each of n types

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

```
$ python3 couponcollectorredux.py 1000
6276
$ python3 couponcollectorredux.py 1000
7038
$ python3 couponcollectorredux.py 1000000
13401736
$ _
```

Defining Functions

Defining Functions

couponcollectorredux.py

```
1  import stdarray
2  import stdio
3  import stdrandom
4  import sys
5
6  def main():
7      n = int(sys.argv[1])
8      stdio.writeln(_collect(n))
9
10 def _collect(n):
11     count = 0
12     collectedCount = 0
13     isCollected = stdarray.create1D(n, False)
14     while collectedCount < n:
15         value = _getCoupon(n)
16         count += 1
17         if not isCollected[value]:
18             collectedCount += 1
19             isCollected[value] = True
20     return count
21
22 def _getCoupon(n):
23     return stdrandom.uniformInt(0, n)
24
25 if __name__ == '__main__':
26     main()
```

Defining Functions

Defining Functions

Program: `playthattunedeluxe.py`

Defining Functions

Program: `playthattunedeluxe.py`

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

Defining Functions

Program: `playthattunedeluxe.py`

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

Defining Functions

Program: `playthattunedeluxe.py`

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

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

Defining Functions

Program: `playthattunedeluxe.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
```

Defining Functions

Program: `playthattunedeluxe.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
$ _
```

Defining Functions

Program: `playthattunedeluxe.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 playthattunedeluxe.py < ../data/elise.txt
```

Defining Functions

Program: `playthattunedeluxe.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 playthattunedeluxe.py < ../data/elise.txt
$ _
```

Defining Functions

Defining Functions

playthattunedeluxe.py

```
import math
import stdarray
import stdaudio
import stdio

def main():
    while not stdio.isEmpty():
        pitch = stdio.readInt()
        duration = stdio.readFloat()
        stdaudio.playSamples(_createRichNote(pitch, duration))
    stdaudio.wait()

def _createRichNote(pitch, duration):
    NOTES_ON_SCALE = 12
    CONCERT_A = 440.0
    hz = CONCERT_A * math.pow(2, pitch / NOTES_ON_SCALE)
    mid = _createNote(hz, duration)
    hi = _createNote(2 * hz, duration)
    lo = _createNote(hz / 2, duration)
    hiAndLo = _superpose(hi, lo, 0.5, 0.5)
    return _superpose(mid, hiAndLo, 0.5, 0.5)

def _createNote(hz, duration):
    SPS = 44100
    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)
    return note

def _superpose(a, b, aWeight, bWeight):
    c = stdarray.create1D(len(a), 0.0)
    for i in range(len(a)):
        c[i] = a[i] * aWeight + b[i] * bWeight
    return c
```

Defining Functions

playthattunedeluxe.py

```
36
37 if __name__ == '__main__':
38     main()
```

Filter, Lambda, Map, and Reduce Functions

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Example

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

```
>>> _
```

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Example

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

```
>>> primes = filter(is_prime, range(11))
```

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Example

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

```
>>> primes = filter(is_prime, range(11))
>>> _
```

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Example

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

```
>>> primes = filter(is_prime, range(11))
>>> list(primes)
```

Filter, Lambda, Map, and Reduce Functions

Functions in Python are first-class objects, meaning they can take functions as arguments and return functions as results

`filter(f, seq)` returns those items of `seq` for which `f(item)` is `True`

Example

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

```
>>> primes = filter(is_prime, range(11))
>>> list(primes)
[2, 3, 5, 7]
>>> _
```

Filter, Lambda, Map, and Reduce Functions

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Example

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

```
>>> _
```

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Example

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

```
>>> odds = filter(lambda x : x % 2 != 0, range(11))
```

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Example

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

```
>>> odds = filter(lambda x : x % 2 != 0, range(11))
>>> _
```

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Example

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

```
>>> odds = filter(lambda x : x % 2 != 0, range(11))
>>> list(odds)
```

Filter, Lambda, Map, and Reduce Functions

A lambda function is a "disposable" function that we can define just when we need it and then immediately throw it away after we are done using it

Example

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

```
>>> odds = filter(lambda x : x % 2 != 0, range(11))
>>> list(odds)
[1, 3, 5, 7, 9]
>>> _
```

Filter, Lambda, Map, and Reduce Functions

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Example

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

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Example

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

```
>>> squares = map(lambda x : x ** 2, range(11))
```

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Example

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

```
>>> squares = map(lambda x : x ** 2, range(11))
>>> _
```

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Example

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

```
>>> squares = map(lambda x : x ** 2, range(11))
>>> list(squares)
```

Filter, Lambda, Map, and Reduce Functions

`map(f, seq)` returns a list of the results of applying the function `f` to the items of `seq`

Example

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

```
>>> squares = map(lambda x : x ** 2, range(11))
>>> list(squares)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> _
```

Filter, Lambda, Map, and Reduce Functions

Filter, Lambda, Map, and Reduce Functions

`functools.reduce(f, seq)` applies the function `f` of two arguments cumulatively to the items of `seq` to reduce the sequence to a single value

Filter, Lambda, Map, and Reduce Functions

`functools.reduce(f, seq)` applies the function `f` of two arguments cumulatively to the items of `seq` to reduce the sequence to a single value

Example

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

```
>>> _
```

Filter, Lambda, Map, and Reduce Functions

`functools.reduce(f, seq)` applies the function `f` of two arguments cumulatively to the items of `seq` to reduce the sequence to a single value

Example

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

```
>>> total = functools.reduce(lambda x, y: x + y, range(11))
```

Filter, Lambda, Map, and Reduce Functions

`functools.reduce(f, seq)` applies the function `f` of two arguments cumulatively to the items of `seq` to reduce the sequence to a single value

Example

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

```
>>> total = functools.reduce(lambda x, y: x + y, range(11))
55
>>> _
```