

# Outline

1 Defining Functions

2 Filter, Lambda, Map, and Reduce Functions



#### Function definition

#### Function definition

#### Return statement

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

#### Function definition

#### Return statement

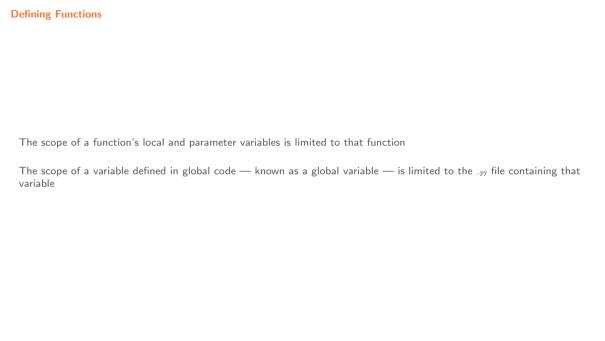
```
return [<expression>]
```

### 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</pre>
```









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

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(i, n + i):
        total += 1 / (i ** r)
    return total
```



# 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)
```

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]
```



 ${\color{red} \textbf{Program:}} \ {\tiny \texttt{harmonicredux.py}}$ 

Program: harmonicredux.py

ullet Command-line input: n (int)

Program: harmonicredux.py

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

Program: harmonicredux.py

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

#### >\_ ~/workspace/ipp/programs

\$\_

Program: harmonicredux.py

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

#### >\_ ~/workspace/ipp/programs

\$ python3 harmonicredux.py 10

 $Program: {\tt harmonic redux.py}$ 

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

#### >\_ ~/workspace/ipp/programs

\$ python3 harmonicredux.py 10
2.9289682539682538

Φ

 $Program: {\tt harmonic redux.py}$ 

- Command-line input: *n* (int)
- Standard output: the *n*th 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

# $Program: {\tt harmonic redux.py}$

- Command-line input: *n* (int)
- Standard output: the *n*th 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
$ _
```

Program: harmonicredux.py

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

- \$ python3 harmonicredux.py 10
- 2.9289682539682538
- \$ python3 harmonicredux.py 1000
- 7.485470860550343
- \$ python3 harmonicredux.pv 10000

# $Program: {\tt harmonic redux.py}$

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

#### >\_ ~/workspace/ipp/program:

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



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

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

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

```
import stdio
import sys

def main():
    n = int(sys.argv[i])
    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()
```

When a program is imported as a library, the program's  $\_\_name\_\_$  attribute is not set to  $\_\_main\_\_$ '

```
>_ "/workspace/ipp/programs
>>> _
```

```
import stdio
import stdio
import sys

def main():
    n = int(sys.argv[i])
    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()
```

When a program is imported as a library, the program's \_\_name\_\_ attribute is not set to \_\_main\_\_'

```
>_ "/workspace/ipp/programs
>>> import harmonicredux
```

```
import stdio
import sys

def main():
    n = int(sys.argv[i])
    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()
```

When a program is imported as a library, the program's  $\_\_name\_\_$  attribute is not set to  $\_\_main\_\_$ '

```
>_ "/workspace/ipp/programs
>>> import harmonicredux
>>> _
```

```
import stdio
import sys

def main():
    n = int(sys.argv[i])
    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()
```

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)
```

```
import stdio
import sys

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

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

if __name__ == '__main__':
    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
>>> _
```



 $Program: {\tt couponcollectorredux.py}$ 

 $Program: \ {\tt couponcollectorredux.py}$ 

ullet Command-line input: n (int)

Program: couponcollectorredux.py

• Command-line input: *n* (int)

• Standard output: number of coupons one must collect before obtaining one of each of n types

Program: couponcollectorredux.py

• Command-line input: *n* (int)

ullet Standard output: number of coupons one must collect before obtaining one of each of n types

>_ ~/workspace/ipp/programs		
\$ _		
1		

Program: couponcollectorredux.py

• Command-line input: *n* (int)

ullet Standard output: number of coupons one must collect before obtaining one of each of n types

#### >\_ ~/workspace/ipp/programs

\$ python3 couponcollectorredux.py 1000

Program: couponcollectorredux.py

• Command-line input: *n* (int)

 $\bullet$  Standard output: number of coupons one must collect before obtaining one of each of n types

#### >\_ ~/workspace/ipp/programs

\$ python3 couponcollectorredux.py 1000
6276

\$ \_

Program: couponcollectorredux.py

• Command-line input: n (int)

 $\bullet$  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

Program: couponcollectorredux.py

• Command-line input: *n* (int)

 $\bullet$  Standard output: number of coupons one must collect before obtaining one of each of n types

# \$ python3 couponcollectorredux.py 1000 6276 \$ python3 couponcollectorredux.py 1000 7038 \$ \_

Program: couponcollectorredux.py

• Command-line input: *n* (int)

 $\bullet$  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
```

Program: couponcollectorredux.py

• Command-line input: *n* (int)

ullet Standard output: number of coupons one must collect before obtaining one of each of n types

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



```
☑ couponcollectorredux.py

import stdarray
import stdio
import stdrandom
import sys
def main():
    n = int(sys.argv[1])
    stdio.writeln(collect(n))
def collect(n):
    count = 0
    collectedCount = 0
    isCollected = stdarray.create1D(n, False)
    while collectedCount < n:
        value = _getCoupon(n)
        count += 1
        if not isCollected[value]:
            collectedCount += 1
            isCollected[value] = True
    return count
def _getCoupon(n):
    return stdrandom.uniformInt(0, n)
if __name__ == '__main__':
    main()
```



Program: playthattunedeluxe.py

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

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

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

~/workspace/ipp/programs
\$ -

#### 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

- 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 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
spython3 playthattunedeluxe.py < ../data/elise.txt</pre>
```

- 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
$ _*</pre>
```



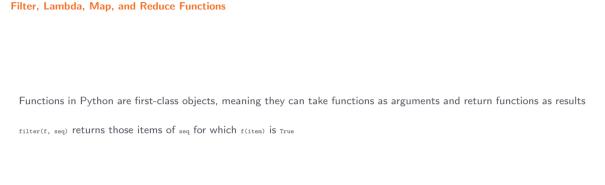


```
📝 playthattunedeluxe.pv
import math
import stdarray
import stdaudio
import stdio
def main().
    while not stdio.isEmpty():
        pitch = stdio.readInt()
        duration = stdio readFloat()
        stdaudio.playSamples(_createRichNote(pitch, duration))
    stdaudio wait()
def createRichNote(pitch, 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 = stdarrav.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 = stdarrav.create1D(len(a), 0.0)
    for i in range(len(a)):
        c[i] = a[i] * aWeight + b[i] * bWeight
    return c
```

```
🕝 playthattunedeluxe.py
if __name__ == '__main__':
    main()
```



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





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

 ${\tt filter(f,\ seq)}$  returns those items of  ${\tt seq}$  for which  ${\tt f(item)}$  is  ${\tt True}$ 

### Example

>\_ ~/workspace/ipp/programs

>>> \_

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

 ${\tt filter(f, seq)}$  returns those items of  ${\tt seq}$  for which  ${\tt f(item)}$  is  ${\tt True}$ 

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

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

 ${\tt filter(f,\ seq)}$  returns those items of  ${\tt seq}$  for which  ${\tt f(item)}$  is  ${\tt True}$ 

```
>> "/workspace/ipp/programs
>>> primes = filter(is_prime, range(11))
>>> _
```

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

 ${\tt filter(f,\ seq)}$  returns those items of  ${\tt seq}$  for which  ${\tt f(item)}$  is  ${\tt True}$ 

```
>> "/workspace/ipp/programs
>>> primes = filter(is_prime, range(11))
>>> list(primes)
```

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

 ${\tt filter(f, seq)}$  returns those items of  ${\tt seq}$  for which  ${\tt f(item)}$  is  ${\tt True}$ 

```
>_ "/workspace/ipp/programs
>>> primes = filter(is_prime, range(11))
>>> list(primes)
[2, 3, 5, 7]
>>> _
```



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 th away after we are done using it	row it

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
>>> _



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

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



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

```
>= "/workspace/ipp/programs
>>> odds = filter(lambda x : x % 2 != 0, range(i1))
>>> _
```



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

```
>= "/workspace/ipp/programs
>>> odds = filter(lambda x : x % 2 != 0, range(i1))
>>> list(odds)
```



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

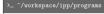
```
>_ "/workspace/ipp/programs
>>> odds = filter(lambda x : x % 2 != 0, range(i1))
>>> list(odds)
[1, 3, 5, 7, 9]
>>> _
```



 $_{map(\texttt{f},\ \texttt{seq})}$  returns a list of the results of applying the function  $_{\texttt{f}}$  to the items of  $_{\texttt{seq}}$ 

 $_{\tt map(f,\ seq)}$  returns a list of the results of applying the function  $_{\tt f}$  to the items of  $_{\tt seq}$ 

### Example



>>> \_

 $_{\tt map(f,\ seq)}$  returns a list of the results of applying the function  $_{\tt f}$  to the items of  $_{\tt seq}$ 

```
>= "/workspace/ipp/programs" >>> squares = map(lambda x : x ** 2, range(i1))
```

 $_{\tt map(f,\ seq)}$  returns a list of the results of applying the function  $_{\tt f}$  to the items of  $_{\tt seq}$ 

```
>_ "/workspace/ipp/programs
>>> squares = map(lambda x : x ** 2, range(11))
>>> _
```

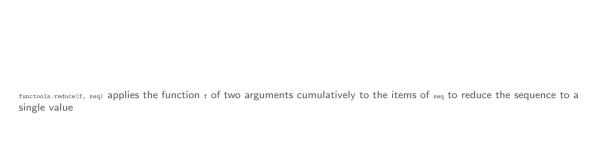
 $_{\tt map(f,\ seq)}$  returns a list of the results of applying the function  $_{\tt f}$  to the items of  $_{\tt seq}$ 

```
>_ "/workspace/ipp/programs
>>> squares = map(lambda x : x ** 2, range(i1))
>>> list(squares)
```

 $_{\tt map(f,\ seq)}$  returns a list of the results of applying the function  ${\tt f}$  to the items of  $_{\tt seq}$ 

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







 $function_{1s,reduce(f, seq)}$  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

>>> \_

 $function_{1s.reduce(f, seq)}$  applies the function f of two arguments cumulatively to the items of seq to reduce the sequence to a single value

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

 $function_{1s.reduce(f, seq)}$  applies the function f of two arguments cumulatively to the items of seq to reduce the sequence to a single value

```
~ /wankana sa /inn /nna sunna
```

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