Funktioner

Daniel Bosk

KTH EECS

15th September 2020

Funktioner

- Som ‘miniprogram’ som går att återanvända.
- Gör att vi kan minimera koduppprepningar.
- Ger färre problem, underlättar underhåll och utbyggnad.

Example (Summera)

```
1 l = [1, 2, 3, 4]
2 print(sum(l))
```

Example (Konkatenera)

```
1 l = ["flag", "pole", "polishing"]
2 print(sum(l))
```

Example (Summera själv)

```
1  summa = 0
2
3  for i in lst:
4      summa += i
```

```
def func(parameters):
    # use {parameters}
    return results
```

Example (summera.py)

```
1  """Our own sum function."""
2
3  def summera(lst):
4      """Sum numbers in lst"""
5      result = 0
6
7      for i in lst:
8          result += i
9
10     return result
11
12
13 ns = [1, 2, 3, 4, 5]
14 print(f"{summera(ns)}")
```

Example (input-int.py)

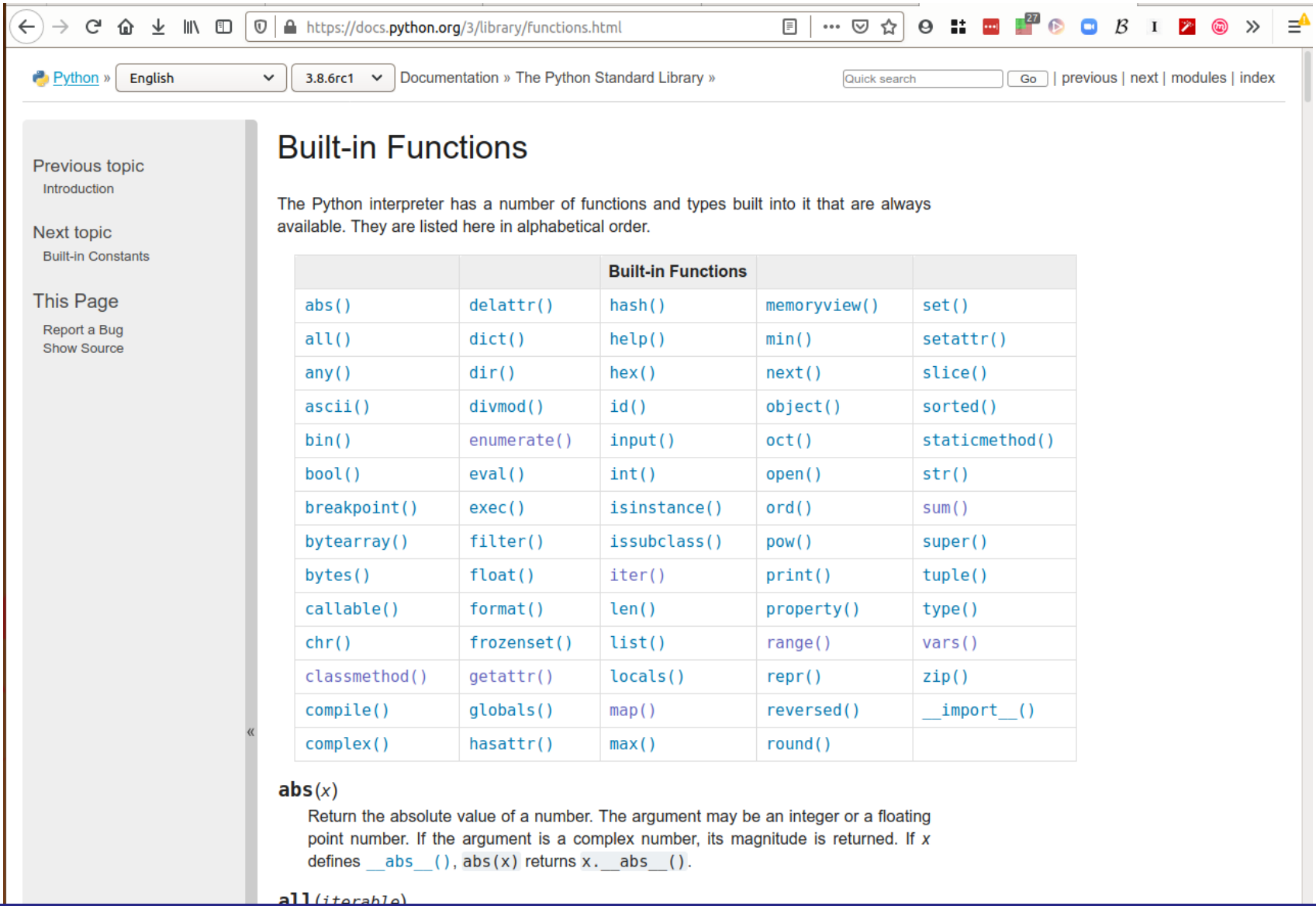
```
1  """Take input more easily."""
2
3  def input_int(prompt):
4      """Take input and convert to int, repeat until success."""
5      while True:
6          try:
7              return int(input(prompt))
8          except ValueError:
9              print(f"Sorry, that's not an integer")
10
11 x = input_int("x = ")
12 y = input_int("y = ")
13
14 print(f"{x} + {y} = {x+y}")
```

Example (input-int-default.py)

```
1  """Take input more easily."""
2
3  def input_int(prompt="Please enter something: "):
4      """Take input and convert to int, repeat if error."""
5      while True:
6          try:
7              return int(input(prompt))
8          except ValueError:
9              print(f"Sorry, that's not an integer")
10
11 x = input_int("x = ")
12 y = input_int()
13
14 print(f"{x} + {y} = {x+y}")
```

Example (input-type.py)

```
1  """Take input more easily."""
2
3  def input_type(t, prompt=""):
4      """Take input and convert to type t, repeat if error."""
5      while True:
6          try:
7              return t(input(prompt))
8          except ValueError:
9              print(f"Sorry, that can't be converted to {t}")
10
11 x = input_type(int, "x = ")
12 y = input_type(int, "y = ")
13
14 print(f"{x} + {y} = {x+y}")
```



https://docs.python.org/3/library/functions.html

Python » English 3.8.6rc1 Documentation » The Python Standard Library »

Quick search Go | previous | next | modules | index

Previous topic
Introduction

Next topic
Built-in Constants

This Page
Report a Bug
Show Source

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

abs(*x*)

Return the absolute value of a number. The argument may be an integer or a floating point number. If the argument is a complex number, its magnitude is returned. If *x* defines `__abs__()`, `abs(x)` returns `x.__abs__()`.

all(*iterable*)

Example (mapping.py, del 1)

```
3  def square(x):
4      """return x squared"""
5      return x**2
6
7  def add_one(x):
8      """return successor of x"""
9      return x+1
10
11 def print_all(lst):
12     """print all elements in lst"""
13     for i in lst:
14         print(i)
```

Example (mapping.py, del 2)

```
17
18  ns = range(10)
19  print_all(ns)
20  print("-"*10)
21
22  ms = list(map(square, ns))
23  print_all(ms)
24  print("-"*10)
25
26  ks = map(add_one, ms)
27  print_all(ks)
28  print("-"*10)
```

Example (filtering.py)

```
1  """Filtering experiments"""
2
3  def gt_two(x):
4      """Return true if x > 2"""
5      return x > 2
6
7
8  ns = range(-10, 10)
9  ms = filter(gt_two, ns)
10
11 for i in ms:
12     print(i, end=" ")
13 print()
```

Example (filter-lambda.py)

```
1  """Filtering experiments"""
2
3  ns = range(-10, 10)
4  ms = filter(lambda x: x > 2, ns)
5
6  for i in ms:
7      print(i, end=" ")
8  print()
```

Example (any-all.py)

```
 7  def check(lst):
 8      gt_two = lambda x: x > 2
 9
10      if any(map(gt_two, lst)):
11          print("There are elements larger than two.")
12      if all(map(gt_two, lst)):
13          print("All elements are larger than two.")
14
15
16  ns = range(-10, 10)
17  print_all(ns)
18  check(ns)
```

Example (any-all.py)

```
20  print("-"*10)
21
22  ms = list(filter(lambda x: x > 2, ns))
23  print_all(ms)
24  check(ms)
```

Example (enum.py)

```
1  """Enumerate objects"""
2
3  l = ["a", "b", "c"]
4
5  for i, c in enumerate(l):
6      print(f"{i}: {c}")
```

Example (zip.py)

```
1  """Enumerate objects"""
2
3  l = ["a", "b", "c"]
4  L = ["A", "B", "C"]
5
6  for v, g in zip(L, l):
7      print(f"{v}: {g}")
```

zip() och enumerate()