A. MicroPython Reference

This appendix provides a reference for some of the key features and functions available in MicroPython. Treat it as a reference resource, and try out some of the functions to see how they work. The REPL is your friend. Try things out. For full details on everything in Python, go to https://microbit-micropython.readthedocs.io/en/latest/.

A.1. Language Contructs

Table A-1 shows some of the functions you can use with numbers.

Table A-1 Language Constructs

Keyword	Description	Example
if	Run a block of code if a condition is true	if a > 10: print("a is greater than 10")

if/else		if a > 10:
		    print("a is greater than 10")
		else:
		    print("a is small")

if/elif/else		if a > 10:
		    print("a is greater than 10")
		elif a > 5:
		    print("a is middle-sized")
		else:
		    print("a is small")

while	Loop while a condition is true	x = 0
		while x < 10:
		    x += 1
		    print(x)

for/in	Loop over the items of a List	for x in range(1, 10): print(x)

try/except	Run uncertain code and trap errors and exceptions	try:
		    do_something_dodgy()
		except:
		    print("Something went wrong")

Keyword	Description	Example
try/except	Run uncertain code and trap errors and exceptions, with access to the exception	try: do_something_dodgy() except Exception as e: print(str(e))
try/finally	Mask and ensure cleanup after exceptions.	try: do_something_dodgy() finally: cleanup()

A.2. Comparisons

To test to see whether two values are the same, we use ==. This is called a *comparison operator*. The comparison operators that we can use are shown in Table A-2.

Table A-2 Comparisons

Comparison	Description	Example
==	Equals	total == 11
!=	Not equals	total != 11
>	Greater than	total > 10
<	Less than	total < 3
>=	Greater than or equal to	total >= 11
<=	Less than or equal to	total <= 2

A.3. Numbers

Table A-3 shows some of the functions you can use with numbers.

Table A-3 Number Functions

Function	Description	Example
abs(x)	Returns the absolute value (removes any minus sign)	**>>>abs(-12.3)** **12.3**
bin(x)	Converts to binary string	**>>> bin(23)** **'0b10111'**
hex(x)	Converts to hexadecimal string	**>>> hex(255)** **'0xff'**
oct(x)	Converts to octal string	**>>> oct(9)** **'0o11'**
round(x, n)	Rounds x to n decimal places	**>>> round(1.111111, 2)** **1.11**
math.log(x)	Natural logarithm	**>>> math.log(10)** **2.302585092994046**
math.pow(x, y)	Raises x to the power y or use x ** y	**>>> math.pow(2, 8)** **256.0**
math.sqrt(x)	Square root	**>>> math.sqrt(16)** **4.0**
math.sin, cos, tan, asin, acos, atan	Trigonometry functions, radians	**>>> math.sin(math.pi / 2)** **1.0**

A.4. Strings

String constants can be enclosed either with single quotes (most common) or with double quotes. Double quotes are useful if you want to include single quotes in the string like this:

s = "It's 3 O'clock"

There are other occasions when you may want to include special characters such as end-of-lines or tabs in a string. To do this, you use what is called *escape characters* that begin with a backslash (\). The only ones that you are likely to need are

- \t—tab character
- \n—new line character

Table A-4 lists some of the strings functions that may be useful.

Table A-4 String Functions

Function	Description	Example
s.endswith(str)	Returns True if the end of the string matches	>>> 'abcdef'.endswith('def') True
s.find(str)	Returns the position of a substring; optional extra arguments of start and end positions to limit the search	>>> 'abcdef'.find('de') 3
s.format(args)	Formats a string using template markers using {}	>>> "Its {0} pm".format('12') "Its 12 pm"
s.isalpha()	Returns True if all the characters are alphabetic	>>> '123abc'.isalpha() False
s.isspace()	Returns True if the character is a space, tab, or other whitespace character	>>> ' \t'.isspace() True
s.lower()	Converts a string into lowercase	>>> 'AbCdE'.lower() 'abcde'
s.replace(old, new)	Replaces all occurrences of "old" with "new"	>>> 'hello world'. replace('world', 'there') 'hello there'
s.split()	Returns a list of all the words in the string separated by spaces; optional parameter to use a different splitting character, the end-of-line character \n being a popular choice	>>> 'abc def'.split() ['abc', 'def']
s.strip()	Removes whitespace from both ends of the string	>>> ' a b '.strip() 'a b'
s.upper()	See lower above.	

A.5. Lists

Table A-5 summarizes the common List functions.

Table A-5 List Functions

Function	Description	Example

Function	Description	Example

del(a[i:j])	Deletes elements from the array from element i to element j-1	>>> a = ['a', 'b', 'c']
		>>> del(a[1:2])
		>>> a
		['a', 'c']

a.append(x)	Appends an element to the end of the List	>>> a = ['a', 'b', 'c']
		>>> a.append('d')
		>>> a
		['a', 'b', 'c', 'd']

a.count(x)	Counts the occurrences of a particular element	>>> a = ['a', 'b', 'a']
		>>> a.count('a')
		2

a.index(x)	Returns the index position of the first occurrence of x in a; optional parameters for start and end index	>>> a = ['a', 'b', 'c']
		>>> a.index('b')
		1

a.insert(i, x)	Inserts x at position i in the List	>>> a = ['a', 'c']
		>>> a.insert(1, 'b')
		>>> a
		['a', 'b', 'c']

a.pop()	Returns the last element of the List and removes it; an optional parameter lets you specify another index position at which to remove	>>> ['a', 'b', 'c']
		>>> a.pop(1)
		'b'
		>>> a
		['a', 'c']

a.remove(x)	Removes the element specified	>>> a = ['a', 'b', 'c']
		>>> a.remove('c')
		>>> a
		['a', 'b']

Function	Description	Example

a.reverse()	Reverses the List	>>> a = ['a', 'b', 'c']
		>>> a.reverse()
		>>> a
		['c', 'b', 'a']

a.sort()	Sorts the List; advanced options when sorting Lists of objects	

A.6. Dictionaries

Table A-6 lists some common Dictionary functions.

Table A-6 Dictionary Functions

Function	Description	Example

len(d)	Returns the number of items in the Dictionary	>>> d = {'a':1, 'b':2}
		>>> len(d)
		2

pop	Deletes an item from the Dictionary	>>> d = {'a':1, 'b':2}
		>>> pop('a')
		1
		>>> d
		{'b': 2}

key in d	Returns True if the Dictionary d contains the key	>>> d = {'a':1, 'b':2}
		>>> 'a' in d
		True

d.clear()	Removes all items from the Dictionary	>>> d = {'a':1, 'b':2}
		>>> d.clear()
		>>> d
		{}

get(key, default)	Returns the value for the key or default if it's not there	>>> d = {'a':1, 'b':2}
		>>> d.get('c', 'c')
		'c'

A.7. Type Conversions

Python contains a number of built-in functions for converting things of one type to another. These are listed in Table A-7.

Table A-7 Type Conversions

Function	Description	Examples

float(x)	Converts x to a floating-point number	>>> float('12.34')
		12.34
		>>> float(12)
		12.0

int(x)	Optional argument to specify the number base	>>> int(12.34)
		12
		>>> int('FF', 16)
		255

list(x)	Converts x to a List; also a handy way to get a List of Dictionary keys	>>> list('abc')
		['a', 'b', 'c']
		>>> d = {'a':1, 'b':2}
		>>> list(d)
		['a', 'b']