Data Types

Data Types		Integer/Float Operations	
Integers	(Whole Numbers)	Addition	+
Floats	(Decimals)	Subtract	-
Strings	(Letters)	Multiply	*
Boolean	(True/False)	Divide	/
None	(Placeholder objects)	Modulo	% (for integers only)
		PEMDAS - Parentheses, Exponents, Multiplication & Division, Addition & Subtraction or "Please Excuse My Dear Aunt Sally"	

Mixing Integers and Floats	None (placeholder)	Booleans	Strings (written characters)
Int (op) int → int	check if something is None via "is"	is equal to ==	Single Quotes ' '
int (op) float → float		is NOT equal to !=	Double Quotes " "
float (op) int → float			Concatenation +
float (op) float → float			Repeat a String *

Casting		Print	Escape Characters	
int()	convert to integer	print()	' \n '	New line
float()	convert to float		' \t '	tab
str()	convert to string		' \' '	Single quote
			' \" '	Double quote
			' \\'	backslash

Variables

Get User Input	Naming Conventions
raw_input()	camelCase snake_case
Variable Assignment	**Variable Reassignment**
x = 1	x = x + 1

Naming Variables	Assignment Operations
Can only contain letters (a–z, A–Z), digits (0–9), and the underscore _ CANNOT begin with numbers CANNOT contain spaces or other symbols CANNOT use Python keywords (ie. True, False, None, is, not, and, or, for, if, elif, else, def) Can only contain letters (a–z, A–Z), digits (0–9), and the underscore _ CANNOT begin with numbers CANNOT contain spaces or other symbols CANNOT use Python keywords (ie. True, False, None, is, not, and, or, for, if, elif, else, def)	+= for example: **x += 1** → **x = x + 1** -= for example: **x -= 2** → **x = x - 2** *= for example: **x *= 3** → **x = x * 3** /= for example: **x /= 4** → **x = x / 4** %= for example: **x += 5** → **x = x + 5**

Conditionals

Comparing Numbers (integers and floats)	
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	is equal to
!=	is not equal to
is	is identical to
is not	is not identical to

Comparing Booleans	
==	is equal to
!=	is not equal to
is	is identical to
is not	is not identical to

Comparing Strings	
==	is equal to
!=	is not equal to
is	is identical to
is not	is not identical to

Boolean Logical Operators

NOT	AND	OR
not True → False not False → True	True and True → True True and False → False False and True → False False and False → False	True or True → True True or False → True False or True → True False or False → False

Boolean Order of Operations
1. () 2. Not 3. <, <=, >, >= ==, !=, is, is not, math operations 4. And 3. Or

Conditional Statements - if / elif / else

Example 1	**Example 2**

if <condition>:
  print(…)
	if <condition>:
	  print(…)
	else:
	  print(…)

Example 3	**Example 4**

if <condition>:
  print(…)
elif:
  print(…)
	if <condition>:
	  print(…)
	elif:
	  print(…)
	else:
	  print(…)

Functions

Defining a Function	Calling a Function
def function_name(): print("Hello World!")	function_name() → "Hello World!"
Return Statement	**Saving a Return**
def get_name(): return "RACECAR"	class = get_name() print(class) → "RACECAR"
Arguments	More Examples
def greet(name): print("Hello " + name)	greet("Daniel!") → "Hello Daniel" greet("Dan") → "Hello Dan"
Multiple Arguments	More Examples
def greet(name1, name2): print("Hello "+name1 +" and "+name2)	greet("Daniel", "Dan") → "Hello Daniel and Dan" greet("Alex", "Wendy") → "Hello Alex and Wendy" greet("Nishanth", "Sabina") → "Hello Nishanth and Sabina"

Calling a Function from a Function

def square(num):
  return num*num
def sum_of_squares(x, y, z):
  return square(x) + square(y) + square(z)

Calling a Function from a Function (con't)
print(sum_of_squares(1, 2, 3)) → 10 print(sum_of_squares(2, 4, 6)) → 56

Data Structures

String Indexing	Example
String indexing starts at 0 (not 1)!!! 	my_str[3] → "E"
Negative Indexing	**Example**
String indexing starts backwards at -1 (not 0)!!! 	my_str = "RACECAR" my_str[-2] → "A"
Slicing	**Example**
<string>[start : stop : step] start (inclusive): start index end (exclusive): end index + 1	 s = "racecar middle school" s[8:15:1] → "middle" s[0:5:2] → "rcc"
Negative Slicing	**Example**
<string>[start : stop : step] start (inclusive): start index end (exclusive): end index + 1	s = "racecar middle school" s[-1:-7:-1] → "loohcs" s[-4:-12:-2] → "hsed"
Slicing Default Values	**Example**
default start: 0 defaut end: string length default step: 1	s = "racecar middle school" s[:] → "racecar midde school" s[: : 2] → "rccrmdl col"
Membership Operators	**String Length**

in for example: "r" in "racecar" → returns True not in for example: "b" not in "racecar" → returns False	len("Hello World") → 11

Lists		
fruits = ["apple", "banana", "pear"]		
	Examples	
Indexing:	fruits[0] →	"apple"
slicing:	fruits[0:2] →	"apple", "banana"
in, not in:	"apple" in ["apple", "banana"] →	True
length:	len(fruits) →	3
Appending (add item to the end of a list)	fruits = ["apple", "banana", "pear"] fruits.append("orange") print(fruits) →	["apple", "banana", "pear", "orange"]
Popping (remove item at specified index)	fruits = ["apple", "banana", "pear"] fruits.pop(2) print(fruits) →	["apple", "banana"]

List Functions	Lists within Lists (examples)
sum(<list_name>) max(<list_name>) min(<list_name>) sorted(<list_name>)	misc = ["apple", 3.14, [1, 2, 3, 4]] print(misc[2]) → [1, 2, 3, 4] print(misc[2][1]) → 2
Adding Lists	**Multiplying Lists**
[0] + [1] → [0, 1]	[0]*4 → [0, 0, 0, 0]

Loops

Iterables (objects that can be indexed or looped over)	Lists within Lists (examples)
String Lists	misc = ["apple", 3.14, [1, 2, 3, 4]] print(misc[2]) → [1, 2, 3, 4] print(misc[2][1]) → 2

While Loops	Examples	Example Result
while <condition>: …	apples = 0 while apples < 3: apples += 1 print(apples)	Prints 1, 2, 3

Infinite Loops	
while True: print("hi")	Prints "hi" forever :(

For Loops	Examples	Example Result
for var in <iterable>: …	for var in [1, 2, 3, 4]: print(var)	Prints 1, 2, 3, 4
	For c in 'banana': print(c)	Prints 'b', 'a', 'n', 'a', 'n', 'a' (a letter a line)

Range()	Examples	Example Result
Create lists with range(): range(start, stop, step)	range(0, 5, 1)	[0, 1, 2, 3, 4]
	range(6, 10, 2)	[6, 8]

More Python Resources

Online Python 3 → Want to practice coding at home? Do it here!
** **Make sure to select "Python 3" in "Language"** **
https://www.onlinegdb.com/

Python Like You Mean It → Want to learn more about python? Here's an AMAZING Python Tutorial!
https://www.pythonlikeyoumeanit.com/

Coding Bat → Practice your newfound python skills here!
https://codingbat.com/python

Stack Overflow → Got coding questions? Other people do too! Ask them here :)
** **Type your question in the "Search" box above!** **
https://stackoverflow.com/