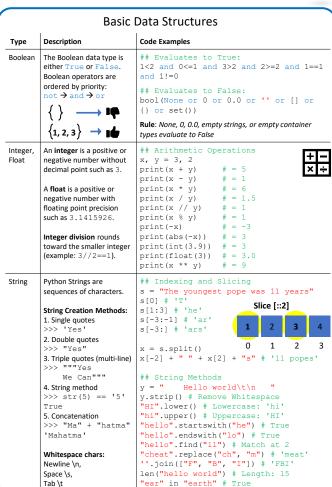


The Ultimate Python Cheat Sheet



	Keywo	ords
Keyword	Description	Code Examples
False, True	Boolean data type	False == (1 > 2) True == (2 > 1)
and, or, not	Logical operators → Both are true → Either is true → Flips Boolean	True and True # True True or False # True not False # True
break	Ends loop prematurely	while True: break # finite loop
continue	Finishes current loop iteration	while True: continue print("42") # dead code
class	Defines new class	<pre>class Coffee: # Define your class</pre>
def	Defines a new function or class method.	<pre>def say_hi(): print('hi')</pre>
if, elif, else	Conditional execution: - "if" condition == True? - "elif" condition == True? - Fallback: else branch	<pre>x = int(input("ur val:")) if x > 3: print("Big") elif x == 3: print("3") else: print("Small")</pre>
for, while	# For loop for i in [0,1,2]: print(i)	<pre># While loop does same j = 0 while j < 3: print(j); j = j + 1</pre>
in	Sequence membership	42 in [2, 39, 42] # True
is	Same object memory location	y = x = 3 x is y # True [3] is [3] # False
None	Empty value constant	print() is None # True
lambda	Anonymous function	(lambda x: x+3)(3) # 6
return	Terminates function. Optional return value defines function result.	<pre>def increment(x): return x + 1 increment(4) # returns 5</pre>



		Complex Data Structures		
Туре	Description	Example	Туре	Desc
List	Stores a sequence of elements. Unlike strings, you	1 = [1, 2, 2] print(len(1)) # 3	Dictionary	Usefu
	can modify list objects (they're mutable).		Reading and	Read speci
Adding elements	Add elements to a list with (i) append, (ii) insert, or (iii) list concatenation.	[1, 2].append(4) # [1, 2, 4] [1, 4].insert(1,9) # [1, 9, 4] [1, 2] + [4] # [1, 2, 4]	writing brace and accee the	
Removal	Slow for lists	[1, 2, 2, 4].remove(1) # [2, 2, 4]		the c
Reversing	Reverses list order	[1, 2, 3].reverse() # [3, 2, 1]	Dictionary You of Iteration value with	
Sorting	Sorts list using fast Timsort	[2, 4, 2].sort() # [2, 2, 4]		
Indexing	Finds the first occurrence of an element & returns index. Slow worst case for whole list traversal.	[2, 2, 4].index(2) # index of item 2 is 0 [2, 2, 4].index(2,1) # index of item 2 after pos 1 is 1	Member- ship operator	Chec set, li an el
Stack	Use Python lists via the list operations append() and pop()	<pre>stack = [3] stack.append(42) # [3, 42] stack.pop() # 42 (stack: [3]) stack.pop() # 3 (stack: [])</pre>	List & set comprehe nsion	List c conci lists.
Set	An unordered collection of unique elements (at-most-once) → fast membership O(1)	<pre>basket = {'apple', 'eggs',</pre>		claus more Set o

Туре	Description	Example
Dictionary	Useful data structure for storing (key, value) pairs	cal = {'apple' : 52, 'banana' : 89, 'choco' : 546} # calories
Reading and writing elements	Read and write elements by specifying the key within the brackets. Use the keys() and values() functions to access all keys and values of the dictionary	<pre>print(cal['apple'] < cal['choco']) # True cal['cappu'] = 74 print(cal['banana'] < cal['cappu']) # False print('apple' in cal.keys()) # True print(52 in cal.values()) # True</pre>
Dictionary Iteration	You can access the (key, value) pairs of a dictionary with the items () method.	<pre>for k, v in cal.items(): print(k) if v > 500 else '' # 'choco'</pre>
Member- ship operator	Check with the in keyword if set, list, or dictionary contains an element. Set membership is faster than list membership.	<pre>basket = {'apple', 'eggs',</pre>
List & set comprehe nsion	List comprehension is the concise Python way to create lists. Use brackets plus an expression, followed by a for clause. Close with zero or more for or if clauses. Set comprehension works similar to list comprehension.	<pre>l = ['hi ' + x for x in ['Alice', 'Bob', 'Pete']] # ['Hi Alice', 'Hi Bob', 'Hi Pete'] 12 = [x * y for x in range(3) for y in range(3) if x>y] # [0, 0, 2] squares = { x**2 for x in [0,2,4] if x < 4 } # {0, 4}</pre>