SKILL<S/ASH>

PYTHON CHEATSHEET

PART 2

SWIPE

SKILL<S/ASH>

Functions And Tricks

Keyword	Description	Code Example	Result
Map(Func, Iter)	Executes the function on all elements of the iterable	list(map(**lambda** x: x[0], ['red','green', 'blue']))	['r', 'g', 'b']
Map(Func, I1, ..., Ik)	Executes the function on all k elements of the k iterables	list(map(**lambda** x, y: str(x) + ' ' + y + 's' , [0, 2, 2], ['apple', 'orange', 'banana']))	['0 apples', '2 oranges', '2 bananas']
String.Join(Iter)	Concatenates iterable elements separated by string	' marries '.join(list(['Alice', 'Bob']))	'Alice marries Bob'
Filter(Func, Iterable)	Filters out elements in iterable for which function returns False (or 0)	list(filter(**lambda** x: **True if** x>17 **else False**, [1, 15, 17, 18]))	[18]
String.Strip()	Removes leading and trailing whitespaces of string	print(" \n \t 42 \t ".strip())	42
Sorted(Iter)	Sorts iterable in ascending order	sorted([8, 3, 2, 42, 5])	[2, 3, 5, 8, 42]
Sorted(Iter, Key=Key)	Sorts according to the key function in ascending order	sorted([8, 3, 2, 42, 5], key=**lambda** x: 0 **if** x==42 **else** x)	[42, 2, 3, 5, 8]
Help(Func)	Returns documentation of func	help(str.upper())	'... to uppercase.'

Keyword	Description	Code Example	Result
Zip(I1, I2, ...)	Groups the i-th elements of iterators i1, i2,... together	list(zip(['Alice', 'Anna'], ['Bob', 'Jon', 'Frank']))	[('Alice', 'Bob'), ('Anna', 'Jon')]
Unzip	Equal to: 1) unpack the zipped list, 2) zip the result	list(zip(*[('Alice', 'Bob'), ('Anna', 'Jon')])	[('Alice', 'Anna'), ('Bob', 'Jon')]
Enumerate(Iter)	Assigns a counter value to each element of the iterable	list(enumerate(['Alice', 'Bob', 'Jon']))	[(0, 'Alice'), (1, 'Bob'), (2, 'Jon')]
Python -M Http.Server <P>	Share files between PC and phone? Run command in PC's shell. <P> is any port number 0–65535. Type < IP address of PC>:<P> in the phone's browser. You can now browse the files in the PC directory.		
Read Comic	import antigravity	Open the comic series xkcd in your web browser	
Zen Of Python	import this	'...Beautiful is better than ugly. Explicit is ...'	
Swapping Numbers	Swapping variables is a breeze in Python. No offense, Java!	a, b = 'Jane', 'Alice' a, b = b, a	a = 'Alice' b = 'Jane'
Unpacking Arguments	Use a sequence as function arguments via asterisk operator *. Use a dictionary (key, value) via double asterisk operator **	**def** f(x, y, z): **return** x + y * z f(*[1, 3, 4]) f(**{'z' : 4, 'x' : 1, 'y' : 3})	13 13

SKILL<S/ASH>

14 Interview Questions

Question	Code	Question	Code
Check If List Contains Integer X	l = [3, 3, 4, 5, 2, 111, 5] print(111 in l) # True	**Get Missing Number In [1...100]**	def get_missing_number(lst): **return** set(range(lst[len(lst)-1])[1:]) - set(l) l = list(range(1,100)) l.remove(50) print(get_missing_number(l)) # 50

Find Duplicate Number In Integer List	def find_duplicates(elements):
	duplicates, seen = set(), set()
	for element in elements:
	if element in seen:
	duplicates.add(element)
	seen.add(element)
	return list(duplicates)
		Compute The Intersection Of Two Lists	**def** intersect(lst1, lst2):
			res, lst2_copy = [], lst2[:]
			for el in lst1:
			if el in lst2_copy:
			res.append(el)
			lst2_copy.remove(el)
			return res

Check If Two Strings Are Anagrams	def is_anagram(s1, s2): **return** set(s1) == set(s2) print(is_anagram("elvis", "lives")) # True	**Find Max And Min In Unsorted List**	l = [4, 3, 6, 3, 4, 888, 1, -11, 22, 3] print(max(l)) # 888 print(min(l)) # -11
Remove All Duplicates From List	lst = list(range(10)) + list(range(10)) lst = list(set(lst)) print(lst) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	**Reverse String Using Recursion**	def **reverse**(string): if len(string)<=1: **return** string **return** reverse(string[1:])+string[0] print(reverse("hello")) # olleh

Question	Code	Question	Code

Find Pairs Of Integers In List So That Their Sum Is Equal To Integer X	def find_pairs(l, x):
	pairs = []
	for (i, el_1) in enumerate(l):
	for (j, el_2) in enumerate(l[i+1:]):
	if el_1 + el_2 == x:
	pairs.append((el_1, el_2))
	return pairs
		Compute The First N Fibonacci Numbers	a, b = 0, 1
			n = 10
			for i in range(n):
			print(b)
			a, b = b, a+b
			# 1, 1, 2, 3, 5, 8, ...

Check If A String Is A Palindrome	def is_palindrome(phrase): return phrase == phrase[::-1] print(is_palindrome("anna")) # True	**Sort List With Quicksort Algorithm**	def qsort(L): if L == []: return [] return qsort([x for x in L[1:] if x< L[0]]) + L[0:1] + qsort([x for x in L[1:] if x>=L[0]]) lst = [44, 33, 22, 5, 77, 55, 999] print(qsort(lst)) # [5, 22, 33, 44, 55, 77, 999]

Use List As Stack, Array, And Queue	# as a list ...
	l = [3, 4]
	l += [5, 6] # l = [3, 4, 5, 6]
	# ... as a stack ...
	l.append(10) # l = [4, 5, 6, 10]
	l.pop() # l = [4, 5, 6]
	# ... and as a queue
	l.insert(0, 5) # l = [5, 4, 5, 6]
	l.pop() # l = [5, 4, 5]
		Find All Permutation S Of String	def get_permutations(w):
			if len(w)<=1:
			return set(w)
			smaller =
			get_permutations(w[1:])
			perms = set()
			for x in smaller:
			for pos in range(0,len(x)+1):
			perm = x[:pos] + w[0] + x[pos:]
			perms.add(perm)
			return perms
			print(get_permutations("nan"))
			# {'nna', 'ann', 'nan'}

NumPy

Name	Description	Example
A.Shape	The shape attribute of NumPy array a keeps a tuple of integers. Each integer describes the number of elements of the axis.	a = np.array([[1,2],[1,1],[0,0]]) print(np.shape(a)) # (3, 2)
A.Ndim	The ndim attribute is equal to the length of the shape tuple.	print(np.ndim(a)) # 2
*****	The asterisk (star) operator performs the Hadamard product, i.e., multiplies two matrices with equal shape element-wise.	a = np.array([[2, 0], [0, 2]]) b = np.array([[1, 1], [1, 1]]) print(a*b) # [[2 0] [0 2]]
Np.Matmul (A,B), A@B	The standard matrix multiplication operator. Equivalent to the @ operator.	print(np.matmul(a,b)) # [[2 2] [2 2]]
Np.Arange ([Start,]Stop, [Step,])	The standard matrix multiplication operator. Equivalent to the @ operator.	print(np.matmul(a,b)) # [[2 2] [2 2]]
Np.Linspace (Start, Stop, Num=50)	Creates a new 1D numpy array with evenly spread elements within the given interval	print(np.linspace(0,10,3)) # [0. 5. 10.]
Np.Average(A)	Averages over all the values in the numpy array	a = np.array([[2, 0], [0, 2]]) print(np.average(a)) # 1.0

Name	Description	Example
<Slice> = <Val>	Replace the <slice> as selected by the slicing operator with the value <val>.	a = np.array([0, 1, 0, 0, 0]) a[::2] = 2 print(a) # [2 1 2 0 2]
Np.Var(A)	Calculates the variance of a numpy array.	a = np.array([2, 6]) print(np.var(a)) # 4.0
Np.Std(A)	Calculates the standard deviation of a numpy array	print(np.std(a)) # 2.0
Np.Diff(A)	Calculates the difference between subsequent values in NumPy array a	fibs = np.array([0, 1, 1, 2, 3, 5]) print(np.diff(fibs, n=1)) # [1 0 1 1 2]
Np.Cumsum(A)	Calculates the cumulative sum of the elements in NumPy array a.	print(np.cumsum(np.arange(5))) # [0 1 3 6 10]
Np.Sort(A)	Creates a new NumPy array with the values from a (ascending).	a = np.array([10,3,7,1,0]) print(np.sort(a)) # [0 1 3 7 10]
Np.Argsort(A)	Returns the indices of a NumPy array so that the indexed values would be sorted.	a = np.array([10,3,7,1,0]) print(np.argsort(a)) # [4 3 1 2 0]
Np.Max(A)	Returns the maximal value of NumPy array a.	a = np.array([10,3,7,1,0]) print(np.max(a)) # 10
Np.Argmax(A)	Returns the index of the element with maximal value in the NumPy array a.	a = np.array([10,3,7,1,0]) print(np.argmax(a)) # 0

Object Orientation Terms

	Description	Example

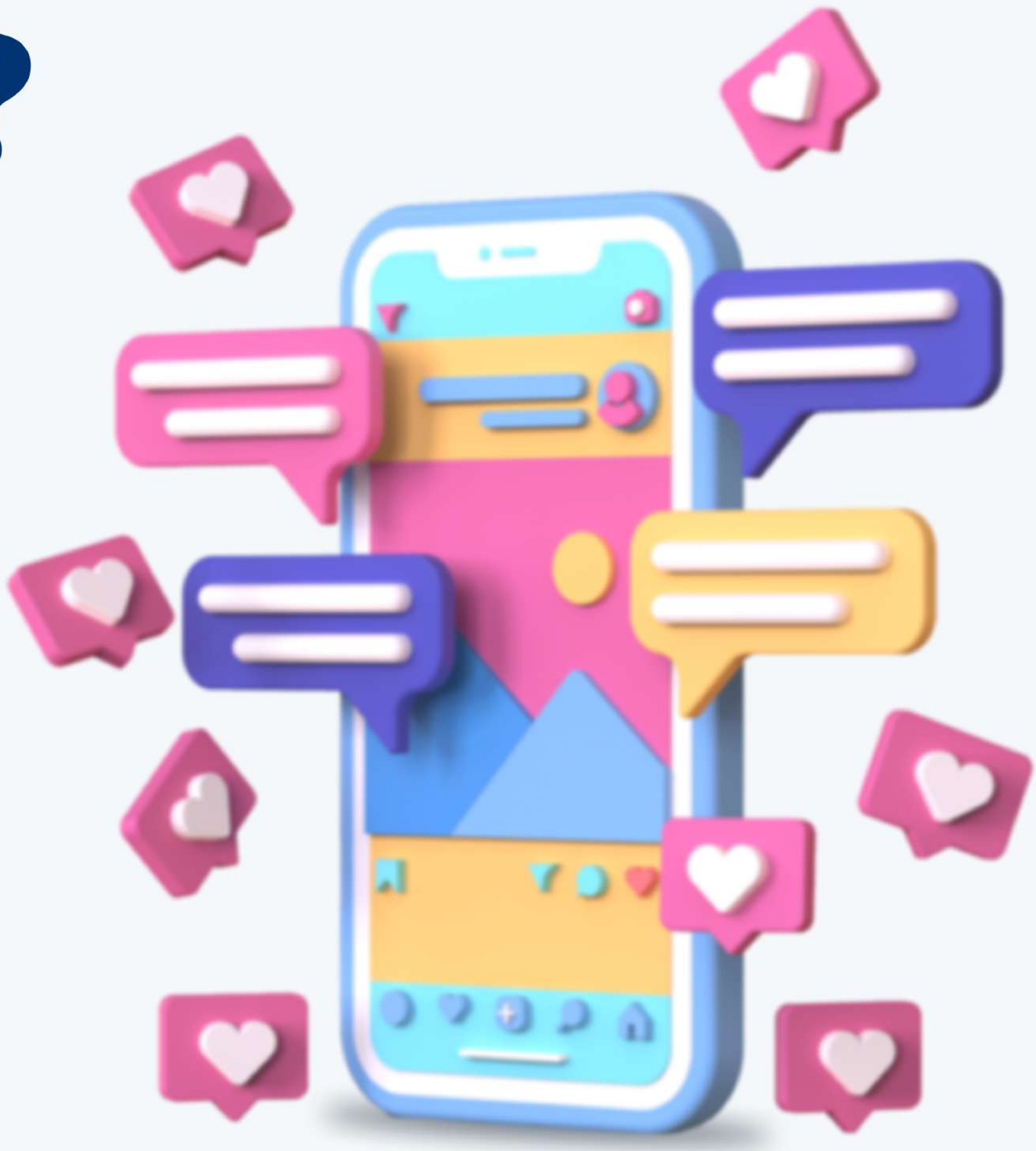
Class	A blueprint to create objects. It defines the data (attributes) and functionality (methods) of the objects. You can access both attributes and methods via the dot notation.	**class** Dog:
		
		# class attribute
		is_hairy = **True**
		
		# constructor
		def __init__(self, name):
		# instance attribute
		self.name = name
		
		# method
		def bark(self):
		print("Wuff")
		
		bello = Dog("bello")
		paris = Dog("paris")
		
		print(bello.name)
		"bello"
		
		print(paris.name)
		"paris"
		
		class Cat:
		
		# method overloading
		def miau(self, times=1):
		print("miau " * times)

Object (=Instance)	A piece of encapsulated data with functionality in your Python program that is built according to a class definition. Often, an object corresponds to a thing in the real world. An example is the object "Obama" that is created according to the class definition "Person". An object consists of an arbitrary number of attributes and methods, encapsulated within a single unit.	
Instantiation	The process of creating an object of a class. This is done with the constructor method __init__(self, ...).	
Method	A subset of the overall functionality of an object. The method is defined similarly to a function (using the keyword "def") in the class definition. An object can have an arbitrary number of methods.	
Self	The first argument when defining any method is always the self argument. This argument specifies the instance on which you call the method. self gives the Python interpreter the information about the concrete instance. To define a method, you use self to modify the instance attributes. But to call an instance method, you do not need to specify self.	

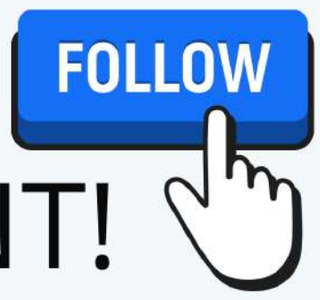
	Description	Example
Encapsulation	Binding together data and functionality that manipulates the data.	fifi = Cat() fifi.miau() "miau "
Attribute	A variable defined for a class (class attribute) or for an object (instance attribute). You use attributes to package data into enclosed units (class or instance).	fifi.miau(5) "miau miau miau miau miau " *# Dynamic attribute* fifi.likes = "mice" print(fifi.likes) "mice" *# Inheritance* **class** Persian_Cat(Cat): classification = "Persian" mimi = Persian_Cat() print(mimi.miau(3)) "miau miau miau "
Class Attribute	(=class variable, static variable, static attribute) A variable that is created statically in the class definition and that is shared by all class objects.	
Instance Attribute (=Instance Variable)	A variable that holds data that belongs only to a single instance. Other instances do not share this variable (in contrast to class attributes). In most cases, you create an instance attribute x in the constructor when creating the instance itself using the self keywords (e.g. self.x = <val>).	
Dynamic Attribute	An instance attribute that is defined dynamically during the execution of the program and that is not defined within any method. For example, you can simply add a new attribute neew to any object o by calling o.neew = <val>.	print(mimi.classification)
Method Overloading	You may want to define a method in a way so that there are multiple options to call it. For example for class X, you define a method f(...) that can be called in three ways: f(a), f(a,b), or f(a,b,c). To this end, you can define the method with default parameters (e.g. f(a, b=None, c=None).	

SKILL<S/ASH>

DID YOU LIKE THIS POST?



TELL US IN THE COMMENTS BELOW!

DROP A 

FOR MORE SUCH VALUABLE CONTENT!