1. isinstance(2, int) • type() method gives you the type, used for chekcking • isinstance() returns bool there works for checking and can be used as statement in code In [1]: 1 = [1,2,3,4,5]1[3] s = 1[2]isinstance(s,int) Out[1]: True lmn = [1, 2, 3, 1, 2, 3]type(lmn) if isinstance(lmn, list): s = set(lmn)print(s) {1, 2, 3} 2. Range is a class in intself In [45]: x = range(6)print(x) print(list(x)) print(type(x)) range(0, 6) [0, 1, 2, 3, 4, 5] <class 'range'> 3. bool() In [46]: | #converts non empty string or non zero numeric value to True bool(5), bool('') #converts empty string or zero numeric value to False bool(0), bool('') bool (234) bool(0) #False bool('') #False bool('') Out[46]: True s = input()if bool(s): print(s) else: print('It is empty') hello hello 4. Format Specifier In [6]: # pi approx = 22 / 7 # print(f'The value of pi is approximately {pi_approx:.2f}') #print('The value of pi is approximately {:.2f}'.format(pi approx)) #print(f'The value of pi is approximately %.2f'%(pi_approx)) #print('{0:5d}'.format(1)) #print('{:5d}'.format(111111)) # :.2f, %.2f pi = 3.146957846393f'{pi:.2f}' print('{}'.format(pi)) print('{:.3f}'.format(pi)) print('{0:5d}'.format(1)) print('{0:5d}'.format(123)) 3.146957846393 3.147 123 5. not > and > or6. Identity and Membership In [7]: # is - object identity #is not - negated object identity # in - membership #not in - negated membership #example1: num = '9874648393' if num[0] in '6849': print(bool('yes')) #first digit can only be one of 6849, verify it 1 = [1, 2, 3, 4, 5, 6, 7, 0]**if** 9 **not in** 1: print('not found') **if** 0 **in** 1: print('found') not found found In [8]: #example2: s = '76544666666' if 6*'6' in s:#6 cannot be repeated more than 5 times continously print('invalid') **if** '9812' **not in** s: print('valid') invalid valid In [9]: #is keyword example 1 = [1, 2, 3]1 notcopy = 1l copy1 = list(1)l copy2 = list.copy 1 copy3 = 1[:]for i in range(1): print(l_notcopy is 1, l_copy1 is 1, l_copy2 is 1, l copy3 is 1) True False False False **#On a SIDE NOTE** # copying a tuple tuple1 = (1, 2, 3)tuple2 = tuple(tuple1) tuple3 = tuple1[:] #tuple4 = tuple1.copy() # does not work for i in range(1): print(tuple1 is tuple2, tuple1 is tuple3) $dict1 = \{1:1,2:2\}$ # copying a dict dict2 = dict(dict1) dict3 = dict1.copy() True True 7. Integer representation of ASCII Code ord('^') char = 'abcdefghijklmnopqrstuvwxyz' for i in char: print(ord(i)) #ord('^') 100 101 102 103 104 105 106 108 109 110 111 112 113 114 116 117 118 119 120 121 8. sizeof # sizeof__() In [14]: **from** random **import** randint 1 = [] for i in range(100): 1.append(randint(1,365)) l. sizeof () tuple(1).__sizeof__() str(1).__sizeof__() Out[14]: 524 In [15]: for i in range(1): print('list size in bytes:',l.__sizeof__()) print('tuple size in bytes:',tuple(1).__sizeof__()) print('string size in bytes:',str(l).__sizeof__()) list size in bytes: 888 tuple size in bytes: 824 string size in bytes: 524 9. Range in for loop #eg1: for i in range(3): print(i) #iterates 3 times upto 3-1 index (0,1,2) 0 1 2 #eg2: y = 'abc' for i in range(len(y)): print(i) 0 1 #eg3: power od step function combines with f string for i in range (1,10,3): print(f'|{i}{i+1}{i+2}', end = '') print('|') |123|456|789| 'h' 'e' 'l' 'l' 'o' # character # left to right (positive indexing) 0 1 2 3 4 # right to left (negative indexing) -5 -4 -3 -2 -1s = 'hello' for i in range(0, len(s)): #starts from 0 goes to index 1 less than len(s) print(s[i]) #print(s[-5])#s[-5:-3:1] #s[-1:-3:-1] е 1 10. enumerate (iterable) string, list, tuple are iterable for i in enumerate('hello'): print(i) (0, 'h') (1, 'e') (2, '1') (3, '1') (4, '0') In [21]: for index, val in enumerate('hello'): print(index, val) 0 h 1 e 2 1 3 1 In [22]: for index, val in enumerate('hello'): print(index) 0 1 2 3 t = 1, 2, 3for index, val in enumerate(t): print(index, val) 0 1 2 3 In [24]: 1 = [1,2,3]for index, val in enumerate(l): print(index, val) 0 1 1 2 11. math library import math from math import trunc, exp, fabs, ceil, floor math.trunc(1.2345) Out[27]: 1 math.exp(1)Out[28]: 2.718281828459045 from math import e, inf, nan, pi math.pi Out[29]: 3.141592653589793 import math 5/0 except Exception as e: print(math.nan, 'error') nan error 11.1 random Library from random import random, randint, choice, choices random() Out[32]: 0.05486655111991212 randint(1,365) #including 1 and 365 Out[33]: 130 #choice() accepts any sequence as input In [34]: #and returns an element chosen at random from this sequence. choice(['a','b','c']) #list Out[34]: 'b' choice(('a','b','c')) #tuple Out[35]: 'a' 11.2 calendar Library from calendar import prmonth, weekday prmonth (2021, 7) July 2021 Mo Tu We Th Fr Sa Su 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 In [38]: weekday(2021,7,4) Out[38]: 6 In [39]: **from** time **import** ctime print(ctime()) Thu Aug 5 00:25:17 2021 12. Built-in functions In [40]: #min(),max(),abs(),round(),help(), id(), ord() lst = [5, 2, 3]print('lst: ', lst) for i in range(1): print('min value: ',min(lst)) print('max value: ',max(lst)) print('sum: ',sum(lst)) print('after sorting: ',sorted(lst)) print('absolute value of -3.567: ',abs(-3.567)) print('3.142858 rounded off to 3 digs: ',round(3.142858,3)) print('memory loc is: ',id(lst)) #help(range) id('a') # This value is the memory address of the object #and will be different every time you run the program chr(97) #The valid range of the integer is from 0 through 1,114,111 ord('a') #opposite of chr lst: [5, 2, 3] min value: 2 max value: 5 sum: 10 after sorting: [2, 3, 5] absolute value of -3.567: 3.567 3.142858 rounded off to 3 digs: 3.143 memory loc is: 140545729149056 Out[40]: 97 In [41]: #sorted also works for string type (unlike l.sort()) #but after sorting it returns as a list as shown below s = 'zxt's new = sorted(s) print(s_new) #however if you want to convert back into string (list to string WOW // how????) print(*s new, sep=',') print(*s new, sep='') print(*s_new, sep=' ') ['t', 'x', 'z'] t,x,z txz 13. Parameter and Arguments variables used in function definition are called parameters values passed to a function call are called arguments Parameters that are assigned a value in the function definition are called default parameters. • Default parameters always come at the end of the parameter list in a function definition. • The argument corresponding to a default parameter is optional in a function call. 14. Call By Value (Inside Fn Vs Outside Fn) Passing arguments like whole numbers, strings or tuples* to a function, you can not change the value of these as they are constant • tuple should be formed using int , float or strings or using immutable objects (HASHABLE) • values (elements) that are changed INSIDE the function, are NOT changed OUTSIDE the function. In [42]: def double(x): x=x*2 return x print(double(a)) print(a) 8 def double(x): In [43]: x**=**x*****2 return x a = 's' print(double('s')) print(a) SS 15. Call By Reference (Inside Fn Vs Outside Fn) • passing mutable objects like list can be considered as call by reference because when their values (elements) are changed INSIDE the function, then it will ALSO be reflected OUTSIDE the function. In [44]: def double(1): return 1.append(100) 1 = [1, 23, 45, 7]print(f'before: {1}') double(1) print(f'after: {1}') print(l) before: [1, 23, 45, 7] after: [1, 23, 45, 7, 100] [1, 23, 45, 7, 100] 16. Create Global Variable inside a function • When we create a variable inside a function, it is local by default. • When we define a variable outside of a function, it is global by default. You don't have to use global keyword. If we do not declare a variable as a global inside function, We can only access the global variable but cannot modify it. • We use global keyword to create a global variable or modify a global variable inside a function. • Local variable takes over global inside the function if the global variable and local variable have the same name Use of global keyword outside a function has no effect. In [45]: def foo(): global x x = 'global'print("x inside:", x) print("x outside:", x) x inside: global x outside: global # #Example1: # def foo(): # # print(x) # UnboundLocalError: local variable 'x' referenced before assignment x = "local" $print(f'x inside foo = \{x\}')$ # # x = "global"# foo() # print(f'x outside foo = {x}') # #Example 2: # def foo(): global x $print(f'x inside foo = \{x\}')$ x=x+1 $print(f'x inside foo = \{x\}')$ # # x = 10# print(f'x outside foo = {x}') # foo() # print(f'x outside foo = {x}') # #Example 3: # def foo(): $print(f'x inside foo = \{y\}') # SyntaxError: name 'x' is used prior to global declaration$ global y y = "global again" # y = "global"# foo() # print(f'x outside foo = {y}') 17. iter() In [46]: patients = ['p01','p02','p03','p04'] doctor = iter(patients) for i in range(5): print(next(doctor)) p02 p03 p04 StopIteration Traceback (most recent call last) <ipython-input-46-57e52426bdcb> in <module> 2 doctor = iter(patients) 3 for i in range (5): print(next(doctor)) StopIteration: 18. Generator In [42]: def foo(limit): x = 0while(x<limit):</pre> yield x* x yield x* x * x x **+=** 1 try: a = foo(4)print(next(a), next(a)) print(next(a), next(a)) print(next(a), next(a)) print(next(a), next(a)) print(next(a), next(a)) except Exception as e: print(e) 0 0 1 1 9 27 In [38]: **def** gen(n): for i in range(n): yield i A = gen(5)print(next(A)) print(next(A)) def gen(n): for i in range(n): yield i A = gen(5)for i in A: print(i) 0 1 2 3 In [41]: def gen(n): for i in range(n): yield i for i in gen(5): print(i) print(type(A)) 0 <class 'generator'> 19. List from Dictionary list_from_dict = list({'one':1,'two':2,'three':3,}.keys()) list_from_dict = list({'one':1,'two':2,'three':3,}.values()) list_from_dict = list({'one':1,'two':2,'three':3,}.items()) list from dict = list({'one':1,'two':2,'three':3,}.keys()) In [48]: list from dict Out[48]: ['one', 'two', 'three'] In [49]: list_from_dict = list({'one':1,'two':2,'three':3,}.values()) list_from dict Out[49]: [1, 2, 3] In [50]: list_from_dict = list({'one':1,'two':2,'three':3,}.items()) list_from_dict Out[50]: [('one', 1), ('two', 2), ('three', 3)] 20. First and Last Element In [51]: L = [10, 20, 30, 40, 50]print(L[0], L[len(L)-1]) #last element = index value @ len(list) - 1 print(L[1:-1]) #exclude first and last element S = '[Mumbai]' # remove the brackets i.e remove first and last char values from Sprint(S[1:-1]) 10 50 [20, 30, 40] Mumbai 21. Multiple list update L = [10, 20, 30, 40, 50]#remove 30 update first 2 elements of the list with 80 and 90 L[1:3] = [80, 90] #multiple update at once print(L) L[0:3] = [80, 90] #updates first 3 index 0,1,2 with 2 values 80, 90 print(L) #value 30 gor deleted [10, 80, 90, 40, 50] [80, 90, 40, 50] 22. Tuple - Immutable, Ordered, Indexed, Iteration possible, Slicing possible "T[im] [Ordered] a [Tuple] from an {Indian} {IT} company and {SLICED} into pieces." • "T[im] [immutable] [Ordered] a [Tuple] from an {Indian}(indexing) {IT}(Iterable) company and {SLICED}(slicing) into pieces." • tuple: Immutable, Ordered, Indexed, Iteration possible, Slicing possible SORTING NOTPOSSIBLE t = 1, 2, 3print(t) #IMMUTABLE $\#t[0] = 5 \ \#TypeError$: 'tuple' object does not support item assignment #INDEXABLE print(t[1]) #ITERABLE for i in t: print(i) #SLICABLE print(t[::-1]) (1, 2, 3)1 2 (3, 2, 1)In [54]: #'UNHASHABLE elements (mutable elements) in a tuple' example t = ([1, 23, 4],)print(t) print(type(t)) print(t[0][1]) t[0][1] = 77 #UNHASHABLE ==> Values can be CHANGED/MANIPULATED print(t) ([1, 23, 4],) <class 'tuple'> ([1, 77, 4],)#'Hashable elements in a tuple' example ==> ITEM ASSIGNEMNT NOT POSSIBLE t = (1, 2, 3)print(t[0]) t[0] = 77 #'tuple' object does not support item assignment Traceback (most recent call last) <ipython-input-55-eda4d066cf5f> in <module> 2 t = (1, 2, 3)**3** print(t[0]) -> 4 t[0] = 77 #'tuple' object does not support item assignment TypeError: 'tuple' object does not support item assignment In [56]: #TUPLE METHODS my_tuple = ('a', 'p', 'p', 'l', 'e',) print(my_tuple.count('p')) print(my_tuple.index('l')) 3 In [57]: #Since indexing is possible ==> slicing is also possible t = (1, 2, 3, 4, 5)print(t[0:3]) (1, 2, 3) 23. Set - hashable, unordered, mutable, add, delete, iterable INDEXING NOT POSSIBLE (since they are unordered, indexing doesnot make any sense) SLICING NOT POSSIBLE SORTING NOT POSSIBLE • Remeber 'HIM' -HashableImmutable • Close Your Eyes, think of the you favorite 'AD & HUM IT' and repeat again, write a SET of all such ads • 'AD HUM IT' - Add Delete Hashable Unordered Mutable Iterable In [58]: #Only Hashable(immutable) elements are allowed $s = \{1, 2, 3\}$ print(s) #print(s[0]) #indexing, slicing not possible{1, 2, 3} In [59]: #unhashable type cannot be elements $s1=\{[1,2,3],[4,5,6]\}$ print(s1) #TypeError: unhashable type: 'list' Traceback (most recent call last) <ipython-input-59-08dc1ffe917f> in <module> 1 #unhashable type cannot be elements $----> 2 s1=\{[1,2,3],[4,5,6]\}$ 3 print(s1) #TypeError: unhashable type: 'list' TypeError: unhashable type: 'list' In [12]: #unordered #mutable #add #delete elements #if you can add/discard/update elements ==> mutable (manipulation is possible) $s2 = {'a', 'b', 123}$ print(s2) s2.add(777) print(s2) s2.discard(777) print(s2) s2.update(['hello','mars']) print(s2) {'b', 123, 'a'} {'b', 777, 123, 'a'} {'b', 123, 'a'} {'hello', 'b', 'a', 123, 'mars'} 24. Dictionary - #DOM Ordered Mutable Hashable(immutable) Keys Sorting Possible my dict = dict([(1,'apple'), (2,'ball')]) # {1: 'apple', 2: 'ball'} print(my dict) dict1 = {'a':1, 'b':2, 'c':3} print(('a',1) in dict1.items()) # copying a dict dict2 = dict(dict1) dict3 = dict1.copy() {1: 'apple', 2: 'ball'} True **#SORTING IS POSSIBLE** $d = \{1:10, 3:30, 5:50, 2:20\}$ print(sorted(d.keys())) print(sorted(d.values())) print(sorted(d.items())) #proof [1, 2, 3, 5] [10, 20, 30, 50] [(1, 10), (2, 20), (3, 30), (5, 50)]In [8]: #duplicate keys $d1 = \{1:23, 2:24, 1:456\}$ Out[8]: {1: 456, 2: 24} In [6]: #All 5 types of possible keys d = {11:123, 1.1:234, 'a':234, (1,2):567, **True**: 6789} Out[6]: {11: 123, 1.1: 234, 'a': 234, (1, 2): 567, True: 6789} #1 and True as keys are same? d = {1:123, 1.1:234, 'a':234, (1,2):567, **True**: 6789} Out[7]: {1: 6789, 1.1: 234, 'a': 234, (1, 2): 567} 24.1 Comparision on a few methods In [3]: a = [1,2,3,4,5]b = (1, 2, 3, 4, 5) $c = \{1, 2, 3, 4, 5\}$ $d = \{1:10, 2:20, 3:30, 4:40, 5:50\}$ print(len(a), min(a), max(a), sum(a)) print(len(b), min(b), max(b), sum(b)) print(len(c), min(c), max(c), sum(c))print(len(d), min(d), max(d), sum(d)) #considers only keys 5 1 5 15 5 1 5 15 5 1 5 15 5 1 5 15 25. Inline Statements • inline if-else inline while loop inline for loop (A { F In }) alias LIST COMPREHENSION

In [62]: #inline if - else a = 34b = 45small = a if a < b else b print(small) #inline while loop (recommended only for small statements) while x>0: print(x); x-=134 3 2 #inline for loop - LIST COMPREHENSION #case1: (A { F In }) a) Append For Loop If/Nested If #case2: (A { IE F}) b) Append If Else For Loop #Model#1: AF (append, for loop) L = [1,2,3,4,5]print([i for i in range(10)]) print([(i, i**2)for i in L]) print([(i,j)for i,j in enumerate(L)]) print([(i+1,j*j)for i,j in enumerate(L)]) d= {1:10, 4:20, 3:19, 6:29 } print([(i, j[0], j[1]) for i, j in enumerate(d.items())])#Model#2: AFI (append, for in list, if statement) fruits = ["apple", "banana", "cherry", "kiwi", "mango"] newlist = [x for x in fruits if "a" in x] print(newlist) lst = [x for x in range(10) if (x<5)]print(lst) #Model3: AIF (append if-else, for loop) newlist3 = [x if x != "banana" else "orange" for x in fruits] print(newlist3) #misq: fruits = ["apple", "banana", "cherry", "kiwi", "mango"] newlist2 = ['hello' for x in fruits] print(newlist2) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)][(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)][(1, 1), (2, 4), (3, 9), (4, 16), (5, 25)][(0, 1, 10), (1, 4, 20), (2, 3, 19), (3, 6, 29)] ['apple', 'banana', 'mango'] [0, 1, 2, 3, 4] ['apple', 'orange', 'cherry', 'kiwi', 'mango'] ['hello', 'hello', 'hello', 'hello'] 26. Nested For Loop In [66]: word = ''.join(sorted(input())) for a in word: for b in word: for c in word: print(a, b, c, sep = '') xhy hhh hhx hhy hxh hxx hyh hyx hyy xhh xhx xhy xxh XXX xxy xyh хух хуу yhh yhx yhy yxh ухх λхλ yyh уух УУУ #row and column swap logic for a matrix In [43]: In [67]: # 1,2,3 # 4,5,6 # 7,8,9 #row # 7,8,9 # 4,5,6 # 1,2,3 n = 3for i in range(n): for j in range(n): print(i,j, '-' ,n-i-1,j) #out[i][j] = mat[n - i - 1][j]0 0 - 2 0 0 1 - 2 1 0 2 - 2 2 1 0 - 1 0 1 1 - 1 1 1 2 - 1 2 2 0 - 0 0 2 1 - 0 1 2 2 - 0 2 In [68]: # 1,2,3 # 4,5,6 # 7,8,9 # #column # 3,2,1 # 6,5,4 # 9,8,7 for i in range(n): for j in range(n): print(i,j, '-',n-j-1,i) #out[i][j] = mat[n - i - 1][j]0 0 - 2 0 0 1 - 1 0 0 2 - 0 0 1 0 - 2 1 1 1 - 1 1 1 2 - 0 1 2 0 - 2 2 2 1 - 1 2 2 2 - 0 2 27. lambda functions writing functions without function name also known as anonymous functions using lambda function you are creating varaibles which are of type function In [69]: x = 5y = 10add = lambda x, y: x+ysub = lambda x, y: x-ymul = lambda x, y: x*ydiv = lambda x, y: x/yprint(add, sub, mul, div) print(type(add)) result = add(10,20)print(result) print(sub(10,20)) <function <lambda> at 0x7fd35a470af0> <function <lambda> at 0x7fd35a545dc0> <function <lambda> at 0x7fd35a545f7 0> <function <lambda> at 0x7fd35a545e50> <class 'function'> -1028. zip function msg = ['hello', 'mars', 'we', 'are', 'coming'] size = [5, 4, 2, 3, 6]print(type(zip(msg, size))) <class 'zip'> In [72]: print((zip(msg, size))) <zip object at 0x7fd35a55c280> In [73]: print(list(zip(msg, size))) #list of tuples from corresponding elements from 2 lists [('hello', 5), ('mars', 4), ('we', 2), ('are', 3), ('coming', 6)] In [74]: print(dict(zip(msg, size))) #2 lists --->into a dictionary 🤌 🤣 🤣 {'hello': 5, 'mars': 4, 'we': 2, 'are': 3, 'coming': 6} In [75]: print(tuple(zip(msg, size))) (('hello', 5), ('mars', 4), ('we', 2), ('are', 3), ('coming', 6)) In [76]: print(set(zip(msg, size))) #the out proves the set is in unordered entity 🤪 {('we', 2), ('are', 3), ('mars', 4), ('coming', 6), ('hello', 5)} 29. map() • map function takes 2 different types of parameters a) function name b) parameters of that function • b)parameters of that function in the below example is a list --> the value gets reflected as it interates over each element a = [10, 20, 30, 40, 50]b = [1, 2, 3, 4, 5]**def** sub (x, y): return x - y c = map(sub, a,b) #example - 1print(c) print(list(c)) def incr(x): return x+1 d = map(incr, a) #example - 2print(list(d)) <map object at 0x7fd35a4b7dc0>[9, 18, 27, 36, 45] [11, 21, 31, 41, 51] In [26]: #example of map menthod in combination with lambda function L = [1, 2, 3, 4, 5, 6]print(list(map(lambda i:i*i, L))) [1, 4, 9, 16, 25, 36] 30. filter() In [33]: L = [1,2,3,4,5,6]print(list(filter(lambda i: i>3, L))) [4, 5, 6] Some Cool Tricks: (ROUTINES) 1. print(*L, sep=", ") The only downside of this is - you can only print it cannot be saved to a variable • to save it to a variable use the join method as below: L=[1,2,3,4,5,6,5,4,3,2,1,0,-1,2.1,3.2,4.0001,-5.3]M=["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"] print(*L, sep=", ") print(*M, sep=", ") 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 0, -1, 2.1, 3.2, 4.0001, -5.3 Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec ##Example1: words = ['this', 'is', 'false'] # a list of words s = '' for word in words: s += word + ' ' # appends words to string variable s print(s) # a string with multiple words #All the above in 1 line of code as below: print(*words, sep=' ') this is false this is false In [92]: ##Example2: words = ['this', 'sentence', 'is', 'false'] s = ' '.join(words)print(s) #All the above in 1 line of code as below: print(*words, sep=' ') this sentence is false this sentence is false 2. Create Matrix from user input In [93]: dim = 3 #can also be made user input mat = [] for i in range(dim): mat.append([]) #initializing an empty matrix for num in input().split(): #space seperated input mat[-1].append(int(num)) print(mat) 1 2 3 4 5 6 7 8 9 [[1, 2, 3], [4, 5, 6], [7, 8, 9]] 3. Unpack the user input to variables string list tuple range In [94]: eq1 = [int(word) for word in input().split()] x, y, z = eq1print(eq1) print(x) print(y) print(z) 12 34 56 [12, 34, 56] 12 34 56 # more unpacking of sequence type collections 11, 12, 13, 14 = 'good' # string num1, num2, num3 = [1, 2, 3] # listb1, b2 = (True, False) # tuple x, y, z = range(3) # rangeprint(11,12,13,14) print(num1, num2, num3) print(b1,b2) print(x,y,z) good 1 2 3 True False 0 1 2 4. List/Tuple with str elements to String In [96]: 1 = ['1', '2', '76']x = ''.join(1)print(x) print(*1,sep='') 1276 1276 In [97]: t = ('1', '2', '76')x = ''.join(t)print(x) 1276 $d = \{ '1': 123, '2': 456, '76': 890 \}$ x = ''.join(d)print(x) 1276 $d = \{ '1': 123, '2': 456, '76': 890 \}$ x = ''.join(d.keys()) #same as above code d.items(), d.values() throws error print(x) 1276 5. Sort a String In [100... l = ['c','d','a'] x = ''.join(sorted(1))print(x) In [101... x = ''.join(sorted(input()))]print(x) hello ehllo 6. Read Each Line from a File row = [int(num) for num in line.strip().split(',')] f = open('test.txt') lines = f.readlines() print(lines) ['1,2,3,4,45\n', '34,45,6,7,7,8,9,9,0'] f = open('test.txt') lines = f.readlines() lst = []for line in lines: row = [int(num) for num in line.strip().split(',')] lst.append(row) print(lst) print(*lst[0], sep=',') [[1, 2, 3, 4, 45], [34, 45, 6, 7, 7, 8, 9, 9, 0]]1,2,3,4,45 **MiniPacks** 1. **OOPS** #Example of Hierarichal Inheritance class Person: #Parent Class def __init__(self, a, b): self.name = aself.age = bdef display(self): print(self.name, self.age) class Student(Person): #Child 1 Class def __init__(self,a,b,c): super().__init__(a,b) self.marks = cdef display(self): super().display() print(self.marks) class Employee(Person): #Child 2 Class def __init__(self, a,b,c): super().__init__(a,b) self.salary = c def display(self): super().display() print(self.salary) s1 = Student('venu', 35, 160) s1.display() e1 = Employee('priyam', 56, 50_000) e1.display() venu 35 priyam 56 50000 class Foo: def a method(self): print('calling from Foo') class Bar(Foo): def a method(self): print('calling from Bar') class FooBar(Foo): def a method(self): super().a method() print('calling from FooBar as well') $obj_1 = Foo()$ obj_1.a_method() $obj_2 = Bar()$ obj_2.a_method() $obj_3 = FooBar()$ obj_3.a_method() calling from Foo calling from Bar calling from Foo calling from FooBar as well 1.1 On the Fly Object attributes can be created ON THE FLY class Foo: def init _(self): print('Foo object created') obj = Foo()obj.x = 10print(obj.x) Foo object created class Foo: x**=**0 obj = Foo()print(obj.x) obj.x = obj.x + 10print(obj.x, Foo.x) 10 0 Some important lessons from above 2 code blocks(1.1): • A class attribute's state can never be explicitly altered by an object. However, it can be changed by using methods. • A class attribute can be referenced by an object, provided there is no object attribute with the same name. • If the same variable name is shared by a class attribute and an object attribute, while trying to access attributes using an object, Python gives precedence to the object attribute. 2. Star Variable In [109.. def foo(a, b): print(a, b) def bar(*a): print(a) def foobar(a, b, *c): print(a, b, c) lst = [3, 4]foo(1, 2)foo(*lst) bar(lst) bar(*lst) bar() bar(1, 2) bar(1, 2, 3, 4) bar(1, 2, *1st) bar(1, 2, 1st) foobar(1, 2, 3, 4, 5) foobar(1, 2)foobar(*lst) foobar(1, 2, *lst) foobar(1, 2, 3, *1) 1 2 3 4 ([3, 4],) (3, 4)() (1, 2) (1, 2, 3, 4) (1, 2, 3, 4) (1, 2, [3, 4]) 1 2 (3, 4, 5) 1 2 () 3 4 () 1 2 (3, 4) 1 2 (3, 'c', 'd', 'a') In []: ## 3. Writing to a File In []: f = open('file.txt', 'w') while True: line = input() **if** line == '': break f.write(line+'\n') f.close() Selected Problems from the Course Recommended method: Read the Question and without looking solution just keep writing the code 🤚 🐫 🤚 The main goal(s) of these selected problems from the course are the following: 1. Fixing time and practising in one go will act as warm up 2. Warm up before an exam 3. Warm up of Python after short or long gap of non-use of Python 4. To increase confidence/morale 5. To develop programming intuition Week 2 "Problem 2: Find whether the given number ends with 0 or 5 or any other number" # num = int(input('Enter a number: ')) # if(num % 5 == 0): # if(num % 10 == 0): # print('0') # else:
print('5') # else: # print('Other') #alternative method num = input() if num[len(num)-1] in '05': print('end with either 0 or 5') print(num[len(num)-1]) else: print('any other number') 345678908 any other number Week 3 "Problem 1: Find the factorial of the given number" In [113... #n = int(input())n = 5# 5 * 4 * 3 * 2 * 1 # 1 * 2 * 3 * 4 * 5 mul = 1for i in range(1,6): #using for loop mul *= i print(mul) 120 In [117... n = 5mul = 1while(n>0): #using while loop mul = mul * n n = n-1print(mul) "Problem 2: Find the number of digits in the given number" In [118... num = abs(int(input('Enter a number: '))) digits = 1 while(num > 10): num = num // 10 digits = digits + 1 print(digits) Enter a number: 12345 In [121... #method2: num = abs(int(input('Enter a number: '))) num str = str(num)print(len(num_str)) Enter a number: 12345 "Problem 3: Reverse the digits in the given number" In [123... | #method 1 num = int(input('Enter a number: ')) absNum = abs(num)rev = absNum % 10 absNum = absNum // 10while(absNum > 0): r = absNum % 10 absNum = absNum // 10 rev = rev * 10 + r **if**(num > 0): print(rev) else: print(rev - 2 * rev) Enter a number: 12345 54321 In [127... #method -2 num = str(abs(int(input()))) print(int(num[::-1])) 12345 54321 "Problem 4: Find whether the entered number is palindrome or not" In [128... #method - 1 num = int(input('Enter a number: ')) absNum = abs(num)rev = absNum % 10 absNum = absNum // 10while (absNum > 0): r = absNum % 10 absNum = absNum // 10 rev = rev * 10 + r **if**(num < 0): rev = rev - 2 * rev if(rev == num): print('Palindrome') else: print('Not a palindrome') Enter a number: 1234321 Palindrome #method -2 n = input()if n.isdigit(): **if** n[::] == n[::-1]: print('palindrome') else: print('not palindrome') 1234321 palindrome "Problem 1: Find all prime numbers less than the entered number" num = int(input('Enter a number: ')) **if**(num > 2): print(2, end = ' ')for i in range(3, num): flag = False for j in range(2, i): **if**(i % j == 0): flag = False break else: flag = True if(flag): print(i, end = ' ') Enter a number: 67 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 #method - 2 def isPrime(n): for i in range(2, n): **if** n%i == 0: return False return True def primeNums(n): result = [] for i in range(2,n): if isPrime(i): result.append(i) return result n = int(input('Enter a number: ')) print(*primeNums(n)) Enter a number: 67 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 "Problem 4: Find the length of longest word from the set of words entered by the user" #method - 1 word = input('Enter a word: ') maxLen = 0while (word != '-1'): count = 0 for letter in word: count = count + 1if(count > maxLen): maxLen = count word = input('Enter a word: ') print('The length of longest word is %s' %maxLen) Enter a word: hello Enter a word: mars Enter a word: we Enter a word: are Enter a word: coming Enter a word: -1 The length of longest word is 6In [159... | # my code print('input a word and press enter to input second word') print('input -1 if you have finished entering words') def wordLen(word): return len(word) word = input() len word = [] n = []while (word != str(-1)): len word.append(word) n.append(wordLen(word)) word = input() print(max(n)) input a word and press enter to input second word input -1 if you have finished entering words hello mars we are coming -1 Week 4 #Birthday Paradox from random import randint n = 30lst = []for i in range(n): lst.append(randint(1,365)) lst.sort() print(lst) i = 0 flag = False while(i<(len(lst)-1)):</pre> **if** lst[i] == lst[i+1]: print('Repeats', lst[i]) flag = True i = i+1if not flag: print('Doesnot Repeat') [10, 34, 92, 100, 117, 135, 135, 152, 159, 175, 176, 201, 202, 227, 238, 251, 259, 261, 262, 293, 310, 315, 32 2, 323, 328, 345, 347, 356, 361, 361] Repeats 135 Repeats 361 In [194... #obvious sort from random import randint lst = [] for i in range(25): lst.append(randint(1,100)) #print(lst) #Let us assume the FIRST ELEMENT is of least Value #Loop over rest of the elements - compare with each element #if any element is least, asign it as min value and append it to a new list #after appending remove that element #in the end your new least is an obvious sorted list X = []while len(lst)>0: min = lst[0]for i in range (len(lst)): if lst[i] < min:</pre> min = lst[i]x.append(min) lst.remove(min) print(x) print(lst) [2, 6, 8, 28, 29, 34, 36, 36, 37, 42, 45, 57, 61, 64, 69, 78, 78, 79, 80, 81, 86, 86, 90, 91, 91] Week 5 "Problem 3: Write a Python code using functions which checks whether the input coordinates form a triangle or not" '''Approach 1: Using distance between points''' def distance(xi, yi, xj, yj): return((((xj - xi) ** 2) + ((yj - yi) ** 2)) ** 0.5) def isTriangle(max, a, b): **if**((a + b) > max): print('\nTriangle') else: print('\nNot a triangle') x1 = float(input('Enter x coorinate of 1st point: ')) y1 = float(input('Enter y coorinate of 1st point: ')) x2 = float(input('\nEnter x coorinate of 2nd point: ')) y2 = float(input('Enter y coorinate of 2nd point: ')) x3 = float(input('\nEnter x coorinate of 3rd point: ')) y3 = float(input('Enter y coorinate of 3rd point: ')) d1 = distance(x1, y1, x2, y2)print(f'\nDistance between points ($\{x1\}$, $\{y1\}$) and ($\{x2\}$, $\{y2\}$) = $\{d1\}$ ') d2 = distance(x2, y2, x3, y3)print(f'\nDistance between points ($\{x2\}$, $\{y2\}$) and ($\{x3\}$, $\{y3\}$) = $\{d2\}$ ') d3 = distance(x3, y3, x1, y1)print(f'\nDistance between points ($\{x3\}$, $\{y3\}$) and ($\{x1\}$, $\{y1\}$) = $\{d3\}$ ') if(d1 > d2): if(d1 > d3):isTriangle(d1, d2, d3) isTriangle(d3, d1, d2) elif(d2 > d3):isTriangle(d2, d1, d3) isTriangle(d3, d1, d2)

s2 = slo print(f' if(s1 != prin else: prin Enter x of Enter y of	2}) = {s1}') 3}) = {s2}')				
Enter y of Enter y of Slope of	coorinate of 2nd properties the line connect.	<pre>point: 4 point: 5 point: 6 ing points (1.0,</pre>			