۱.How are tuples different from lists?

In Python, lists and tuples are declared in different ways. A list is created using square brackets [] whereas the tuple is created using parenthesis ().Lists are *mutable* while tuples are *immutable*, and this marks **the key difference** between the two.

2. How do tuples support the indexing operation ([]) differently from lists? We can convert a tuple to a list using the list function and the tuble function performs thereverse conversion . it is [ ] included in the list of elelments but in touble prameters are in parantez .

3. Are tuples mutable or immutable? Immutable

4. Are the elements in tuples ordered or unordered? Ordered

5. Rewrite the last assignment statement in the following interactive sequence so that it behaves identically but uses tuple unpacking instead of tuple slicing.

```
>>> a = 1, 2, 3, 4, 5, 6, 7, 8
>>> a
(1, 2, 3, 4, 5, 6, 7, 8)
>>> s = a[2:6]
>>> s
(3, 4, 5, 6)
```

a = 1,2,3,4,5,6,7,8

a

(1,2,3,4,5,6,7,8)

S= - ,- ,*s,-,- = a

S=tuple(s)

print (tuple(s))

6. Rewrite the last assignment statement in the following interactive sequence so that it behaves identically but uses tuple slicing instead of tuple unpacking.

```
>>> a = 1, 2, 3, 4, 5, 6, 7, 8
>>> a
(1, 2, 3, 4, 5, 6, 7, 8)
>>> s = _, _, _, *s, _ = a
>>> s = tuple(s)
>>> s
(4, 5, 6, 7)
```

a = 1,2,3,4,5,6,7,8

a

(1,2,3,4,5,6,7,8)

S=a[3:7]

S=tuple(s)

print (tuple(s))

7. Consider the tuple tpl defined as

tpl = 7, 10, -3, 18, 6, 10

Provide one assignment statement that uses tuple unpacking to assign x to the first element and y to the last element.

tpl=7,10,-3,18,6,10

s=x,y = t[0],t[5]

s=tuple(s)

print(tuple(s))

10. Why is a dictionary considered an associative container? A python dictionary is an associative container which permits access based on a key rather than an index a dictionary is called an associative because it associates a key white an item .

11. What statement assigns an empty dictionary to a variable named d? d={}

the contents of a dictionary appear with in curlybace ({}) to access an element whith in a dictionary we use squarebrackets.

12. 12. If d refers to a dictionary, what expression represents the value associated with the key "Fred"?

key []

d['fred'] = 44

in a dictionary every key has an associated value.

d={'fred' :44}

13. What happens when an executing program attempts to retrieve a value using a key that is not present in the dictionary?

A Python KeyError exception is what is raised when you try to access a key that isn't in a dictionary (dict). Python's official documentation says that the KeyError is raised when a mapping key is accessed and isn't found in the mapping. A mapping is a data structure that maps one set of values to another.

14. What happens when an executing program attempts to associate a value with a key that is not present in the dictionary?

A Python KeyError exception is what is raised when you try to access a key that isn't in a dictionary (dict). Python's official documentation says that the KeyError is raised when a mapping key is accessed and isn't found in the mapping. A mapping is a data structure that maps one set of values to another.

15. Are dictionaries mutable or immutable? dictioneries are unordered , mutable, and indexed

16. Given the following dictionary:

d = {3:0, 5:1, 10:1, 8:2, 15:4}

Indicate what each of the following code fragments will print:

(a) print(d)
{3:0 , 5:1 , 10: 1 , 8: 2 , 15 : 4}
(b) for x in d:
print(x)
3,5,10,8,15
(c) for x in d.keys():
print(x)
3,5,10,8,15
(d) for x in d.values():
print(x)
0,1,1,2,4

17. Are the elements in dictionaries ordered or unordered? unordered

20. Explain why the statement

A = {}

does not create an empty set.

to creat an empety set you have to use set() , not{} , the letter creats an empety dictionary

21. Provide the Python statement that assigns the variable A to the empty set ?

to creat an empety set you have to use

A = set ()

22. Are sets mutable or immutable? sets are mutable

23. Given the following initialization statements:

A = {20, 19, 2, 10, 7}

B = {4, 10, 5, 6, 9, 7}

C = {10, 19}

evaluate the following expressions:

(a) A = { 20, 19, 2, 10, 7 }
(b) 20 in A ? True
(c) 20 not in A ? false
(d) A & B ? {7,10}
(e) A | B ? {20,19,2,10,7,4,5,6, 9 }
(f) C < A ? True
(g) C <= A ? True
(h) C <= B ? false
(i) A <= A ? True
(j) A < A ? false
(k) len(A) ? 5
(l) {x + 2 for x in range(10)} ? {2,3,4,5,6,7,8,9,10,11}
(m) {x - 2 for x in A} ? {0,5,8,17,18}
(n) {x - 2 for x in A if x < 10} ? {0,5}

12. For the next set of questions show what each program will print when the user supplies the indicated input text. Write *EXCEPTION* if and when the execution will generate an exception stack trace for an uncaught exception.

(a) print('Begin')

x = int(input())

print(x)

print('End')

i. User enters 22 ⟶ Begin ,22,End

ii. User enters ZZ ⟶ Begin,valuerror

(b) print('Begin')

try:

x = int(input())

print(x)

except ValueError:

print('Wrong!')

print('End')

i. User enters 22 ⟶ Begin , 22,End

ii. User enters ZZ ⟶ Begin , wrong , End

(c) print('Begin')

try: x = int(input())

print(x)

except IndexError:

print('Wrong!')

print('End')

i. User enters 22 ⟶ Begin , 22,End

ii. User enters ZZ ⟶ Begin , valuerror

(d) print('Begin')

```
try: x = int(input())
print(x)
except Exception:
 print('Wrong!')
print('End')
```

i. User enters 22 ⟶ Begin,22,End
ii. User enters ZZ ⟶ Begin , Wrong , End

(e)
```
print('Begin')
try: x = int(input())
 print(x)
except ValueError:
 print('Wrong!')
else:
print('Wow')
print('End')
```

i. User enters 22 ⟶ Begin , 22, wow, End
ii. User enters ZZ ⟶ Begin , wrong,End

(f)
```
print('Begin')
try: x = int(input())
 print(x)
 except ValueError:
 print('Wrong!')
finally:
print('Done')
print('End')
```

i. User enters 22 ⟶ Begin,22, Done, End
ii. User enters ZZ ⟶ Begin , wrong,Done, End

(g)
```
print('Begin')
try: x = int(input())
print(x)
except ValueError:
print('Wrong!')
else:
print('Wow')
 finally:
print('Done')
print('End')
```

i. User enters 22 ⟶ Begin, 22, wow,Done,End
ii. User enters ZZ ⟶ Begin , wrong , Done, End

13. What is the problem with the following code?
try: f() # Function f can raise an exception
except Exception:
print(1)
except ValueError:
print(2)

resents the catch-all handler that can catch any exception not caught by an earlier except block within the try statement. If present, the catch-all except block should be the last except block in the try statement. Since the Exception type matches any exception type, if it appears before another except block, it will intercept a specific exception before a later except block has a chance to see it. This is because a program executes at most one except block when executing a try statement.

14. What is the problem with the following code?
try: f() # Function f can raise an exception
except OSError:
print(1)
except FileNotFoundError:
print(2)

because OSError is more general than FileNotFoundError and PermissionError its except blcok must appear after the except blocksofboth FileNotFoundError and PermissionError. If OSError's except blockappearsin the source earlier, it will catch the file not found and permission error exceptions before the more specific handlers get a chance. The OSError exception type is good to use if you need to defend against all file processingerrorsbutdonotneedthefiner-grainedcontrolofferedbythemorespecificfileexceptiontypes

Thefunctionreturnstrueifcircle circ'sdimensionswouldallowittofitcompletelywithinrectangle rect. If circ is too big, the function returns false. The positions of rect and circ do not influence the result.
11. Consider the following Python code:

```
class Widget:
 def __init__(self, v = 40):
if v >= 40:
self.value = v
else:
self.value = 0
def get(self):
return self.value
def bump(self):
 if self.value < 50:
 self.value += 1
```

def main():
.
.

(a) What does the program print? 41,1

(b) If wid is a Widget object, what is the minimum value the expression wid.get() can return? 0

(c) If wid is a Widget object, what is the maximum value the expression wid.get() can return?41