CSCI 145 · Quiz 6 Reference Sheet (Lectures 11–12)

RECURSION

- **Base case:** answered directly, with no recursive call
- **Recursive case:** calls the function on a smaller input

Input	Smaller input	Typical base case
a number n	`n - 1`	`n == 0` or `n <= 1`
a string word	`word[1:]`	`len(word) == 0`
a list nums	`nums[1:]`	`len(nums) == 0`

PENDING OPERATIONS

A **pending operation** is work done after the call returns

- Work before the call happens on the way down
- Work after the call happens on the way back up, last call first

```
def list_sum(nums):
    if len(nums) == 0:
        return 0
    return nums[0] + list_sum(nums[1:])
```

```
list_sum([4, 7, 2]) = 4 + (7 + (2 + 0)) = 13
```

- In a function that computes and returns an answer (not a turtle drawing), the recursive case must **return** the combined value

TAIL RECURSION OR PENDING OPERATION?

Tail	The call is the last thing done: `return f(nums[1:])`
Pending	Work is left after the call: `return nums[0] + f(nums[1:])`

TWO RECURSIVE CALLS

```
def fib(n):
    if n < 2:
        return n
    return fib(n - 1) + fib(n - 2)
```

The + waits for two calls; smaller calls repeat

THE TURTLE (import turtle as t)

t.forward(distance)	Move forward
t.backward(distance)	Move backward
t.left(angle)	Turn left by angle degrees
t.right(angle)	Turn right by angle degrees

COUNTING WITH A DICTIONARY

```
counts = {}
for item in items:
    if item in counts:
        counts[item] += 1
    else:
        counts[item] = 1
```

The first time: start the count at 1. After that: add 1

TUPLES (IMMUTABLE, LIKE STRINGS)

point = (3, 7)	A tuple of a few related values
point[0]	Indexing and slicing, as for lists
len(point)	Also `len` and `in`, as for lists
point[0] = 5	**TypeError**: a tuple can't change
(5,)	A tuple with one value needs a comma
point = 3, 7	Commas make a tuple (packing)
x, y = point	Unpacking: one name per value, or a **ValueError**
a, b = b, a	Swaps a and b
return x, y	Returns the tuple `(x, y)`
points = [(3, 7)]	A list of tuples
points.append((x, y))	Adds one tuple to the list
for x, y in points:	Unpacks each tuple in the list

DICTIONARIES (MUTABLE, LIKE LISTS)

{}	An empty dictionary
{"Ana": 3, "Ben": 5}	Keys `"Ana"` and `"Ben"` map to 3 and 5
d[key]	The value for key; a missing key is a **KeyError**
d[key] = value	Changes the value for key, or adds a new item
d[key] += 1	Adds 1 to the value for an existing key
key in d	`True` if key is a key in d. Otherwise, `False` (checks keys, not values)
d.get(key, default)	The value for key if key is in d. Otherwise, default
len(d)	The number of items (key: value pairs)
for key in d:	Loops over the keys, in the order they were added
d.items()	`(key, value)` tuples: `for key, value in d.items():`
d.values()	Just the values: `sum(d.values())`

- Keys are unique and immutable: strings, numbers, tuples. A list key is a **TypeError**
- Values can be anything and can repeat