Cheatography

CS1010S Midterms Cheat Sheet

by jinque via cheatography.com/194546/cs/40590/

Types and Operators

/: returns a **float**

%: remainder

int % 10: returns last digit

//: quotient, rounds down to nearest int

int // 10: pops off last digit

int(x): truncates decimals

bool(x): False for x values that are null or empty x. True for otherwise

Slicing

s[start: stop(non-inclusive): step]

s = "12345"

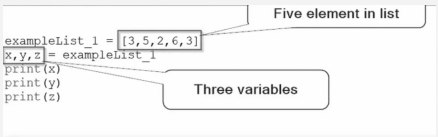
s[0:5:2]	"135"
start and stop must follow the direction as indicated by the step	s[:1:-1] => "-543" s[0:1:-1] => ""
Slicing of tuples **always** returns a tuple	(1, 2, 3)[1: -1] ⇒ (2,)

Comparison

b = (8,9) b = (b, b) # b = ((8, 9), (8, 9)) b in b False

strings are compared lexicographically, if one is not a subset	'bananb' > 'banana' => True
if subset: len(s) is compared	'pqrst' > 'pqrs' => True

ValueError



sum of 1 to 2 ** n

$2^0 + 2^1 + ... + 2^k = 2^{k+1} - 1$

Types of Errors

NameError	Raised when a variable is not found in local or global scope.
IndexError	Raised when index of a sequence is out of range (does not occur for slicing)
ValueError	Raised when a function receives an argument of the correct type but inappropriate value
TypeError	argument passed to a function is incompatible with the type expected / incorrect no. of arguments
SyntaxError	
ZeroDivisionError	
UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable
RecursionError	Infinite looping
RuntimeError	Raised when an error does not fall under any other category.

ValueError

```
int('hello')
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    int('hello')
ValueError: invalid literal for int() with base 10: 'hello'
```

is_fib(n) (T = O(logn), S = O(1))

```
import math
def is_fib(n):
    if n == 0 or n == 1: return True
    a, b = 0,1
    while b < n:
        a, b = b, a + b
        if b == n: return True
    return False
```

Functions on iterables (tuples, strings etc.)

len(), max(), min(), sum()

tuple(*<iterable>*)

<iterable>.count(element)

<iterable>.index(element, start, end)

count_digits

```
def no_of_digits(i):
   if abs(i) < 10:
            return 1
    else:
            return 1 + no_of_ -
dig its(i // 10)
```

factorial

```
def factorial(n):
   if n <= 1: return 1
    else:
            return n * factor -
ial(n - 1)
#T(n), S(n) = O(n)
```

Order of Growth

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

> T = O(2^n)

Indexing/checking element (in)

> T(n) = O(n), S(n) = O(1)

String and Tuple Slicing

> O(n), where n is length of slice (both time and space)

String and Tuple Concatenation

> O(n), where n = len(new seq)

Printing

> T(n) = string: O(n), int: O(1)

T(n) = n * logn

O(n) operations for each level * logn depth

By **jinque**

cheatography.com/jinque/

Not published yet.

Last updated 3rd October, 2023.