****string needs to be in quotes****

Test Case 1: Nothing
input:
expected output:
actual output:
error: needs one string argumentmac:desktop

Test Case 2: String
input: Hello my name is Larry
expected output:
word: Hello
word: my
word: name
word: is
word: Larry
actual output:
word "hello"
word "my"
word "name"
word "is"
word "Larry"

Test Case 3: Comma
input: Hi, I am you're neighbor
expected output:
word: Hi
comma: ,
word: I
word: am
word: you're
word: neighbor
actual output:
word "Hi"
comma ","
word "I"
word "am"
word "you"
single quote "'"
word "re"
word "neighbor"

Test Case 4 Period
input: Mr. Krabbs is a crab
expected output:
word: Mr
period: .
word: Krabbs
word: is
word: a
word: crab
actual output:
word "Mr"
period "."
word "Krabbs"
word "is"
word "a"
word "crab"

Test Case 5: Exclamation Point/Negate
input: Salutations! ! 5
expected output:
word: Salutations
exclamation point: !
exclamation point: !
int: 5
actual output:
word: Salutations
exclamation point: !
exclamation point: !
int: 5

Test Case 6: Colon
input: Food: Burgers and fries.
expected output:
word: Food
colon: :
word: Burgers
word: and
word: fries

period: .
actual output:
word "Food"
unknown character ":"
word "Burgers"
word "and"
word "fries"
period "."

Test Case 7: Semicolon
input: store; too!
expected output:
word: store
semicolon: ;
word: too
exclamation point: !
actual output:
word: store
semicolon: ;
word: too
exclamation point: !

Test Case 8: Question Mark
input: Why?
expected output:
word: Why
question mark: ?
actual output:
word "Why"
unknown character "?"

Test Case 9: Integer
input: I am 21 yrs. old.
expected output:
word: I
word: am
integer: 21
word: yrs
period: .
word: old
period: .

actual output:
word "I"
word "am"
decimal integer "21"
word "yrs"
period "."
word "old"
period "."

Test Case 10: Float
input: pi is: 3.14159
expected output:
word: pi
word: is
colon: :
decimal: 3.14159
actual output:
word "pi"
word "is"
unknown character ":"
float "3.14159"

Test Case 11: Octal Number
input: h
expected output: h
actual output:

Test Case 12: Hex Number
input: 0X234ab, 0x452bC, ox7834ab, 0x3456gh
expected output:
hex number: 0X234ab
comma: ,
hex number: 0x452bC
comma: ,
word: ox
integer: 7834
word: ab
comma: ,
hex number: 0x3456
word: gh

actual output:
hex integer "0X234ab"
comma ","
hex integer "0x452bC"
comma ","
word "ox7834ab"
comma ","
hex integer "0x3456"
word "gh"

Test Case 13: Backslash
input: it is \ me
expected output: word: it
word: is
backslach: \
word: me
actual output:
word "it"
word "is"
backslash "\"
word "me"

Test Case 14: Plus
input: 1 + 2.34 ++ 6 += 0 =+ 1
expected output: integer: 1
addition symbol: +
float: 2.34
increment: ++
int: 6
plus/equals: +=
int: 0
equals: =
addition symbol: +
int: 1
actual output:
decimal integer "1"
plus "+"
float "2.34"
increment "++"
decimal integer "6"
plus equal "+="
decimal integer "0"
equals "="

plus "+"
decimal integer "1"

Test Case 15: Minus
input: 4.56 - 2.3 -- 6 =- 1
expected output: float: 4.56
subtraction symbol: -
float: 2.3
decrement: --
int: 6
equals: =
subtraction symbol: -
int: 1
actual output:
float "4.56"
minus "-"
float "2.3"
decrement "--"
decimal integer "6"
equals "="
minus "-"
decimal integer "1"

Test Case 16: Divide
input: 5 / 67 /= 21 =/ 0
expected output: integer: 5
division symbol: /
integer: 67
divide/equals: /=
int: 21
equals: =
divide: /
int: 0
actual output:
decimal integer "5"
divide "/"
decimal integer "67"
divide equal "/="
decimal integer "21"
equals "="
divide "/"
decimal integer "0"

Test Case 17: Multiplication
input: 65 * 90.6 *= 21 =* 0
expected output:
integer: 65
multiplication symbol: *
float: 90.6
multiplication/equals: *=
int: 21
equals: =
multiplication: *
int: 0
actual output:
decimal integer "65"
multiply or pointer "*"
float "90.6"
multiply equal "*="
decimal integer "21"
equals "="
multiply or pointer "*"
decimal integer "0"

Test Case 18: Left Parenthesis
input: if (not
expected output:
word: if
left parenthesis: (
word: not
actual output:

Test Case 19: Right Parenthesis
input: if) or
expected output: word: if
right parenthesis:)
word: or
actual output:
if keyword "if"
left parenthesis "("
word "not"

***

Test Case 19: Cast / Function
input: (hello))(() helloworld()
expected output:
cast: (hello)
right parenthesis:)
left parenthesis: (
cast: ()
function: helloworld()
actual output:
cast: (hello)
right parenthesis:)
left parenthesis: (
cast: ()
function: helloworld()

Test Case 20: Left Bracket
input: [[,low
expected output:
left bracket: [
left bracket: [
comma: ,
word: low
actual output:
left brace "["
left brace "["
comma ","
word "low"

Test Case 21: Right Bracket
input: it's me]]],
expected output:
word: it's
word:me
right bracket:]
right bracket:]
right bracket:]
comma: ,
actual output:
word "it"
single quote "'"
word "s"
word "me"

right brace "]"
right brace "]"
right brace "]"
comma ","

***

Test Case 20: Array Element
input: [a] []][
expected output:
array element: [a]
array element: []
right bracket:]
left bracket: [
actual output:
array element: [a]
array element: []
right bracket:]
left bracket: [

Test Case 22: Left Curly Brace
input: {//{
expected output:
left curly brace: {
division symbol: /
division symbol: /
left curly brace: {
actual output:
left bracket "{"
comment line "//"
left bracket "{"

Test Case 23: Right Curly Brace
input: } * }
expected output:
right curly brace: }
multiplication symbol: *
right curly brace: }
actual output:
right bracket "}"
multiply or pointer "*"
right bracket "}"

***

Test Case 24: Double Backslash
input: \\ \ , \\
expected output:
double backslash: \\
backslash: \
comma: ,
double backslash: \\
actual output:
backslash "\"
backslash "\"
comma ","
backslash "\"

Test Case 25: Underscore
input: 1_hello
expected output:
integer: 1
underscore: _
word: hello
actual output:
decimal integer "1"
underscore "_"
word "hello"

Test Case 26: Dollar Sign
input: $4.00
expected output:
dollar sign: $
decimal: 4.00
actual output:
dollar sign: $
decimal: 4.00

Test Case 27: Hashtag
input: #famous
expected output:
hashtag: #
word: famous
actual output:
hashtag "#"
word "famous"

Test Case 28: "@"
input: @ me

expected output:
at symbol: @
word: me
actual output:
at sign "@"
word "me"

Test Case 29: Apostraphe
input: ' taking dubs ' it's
expected output:
apostrophe: '
word: taking
word: dubs
apostrophe: '
word: it's
actual output:
single quote "'"
word "taking"
word "dubs"
single quote "'"
word "it"
single quote "'"
word "s"

Test Case 30: Less Than
input: 1 < 3 ; <= 1 =< 0
expected output:
integer: 1
less than symbol: <
integer: 3
semicolon: ;
less than/equal to: 1
equals: =
less than: <
int: 0
actual output:
decimal integer "1"
less than "<"
decimal integer "3"
unknown character ";"
less than or equal "<="
decimal integer "1"
equals "="

less than "<"
decimal integer "0"

Test Case 31: Greater Than
input: 9> 7! >= 0 => 7
expected output:
integer: 9
greater than symbol: >
integer: 7
exclamation point: !
greater than/equal to: >=
int: 0
equals: =
greater than: >
int: 7
actual output:
decimal integer "9"
greater than ">"
decimal integer "7"
not "!"
greater than or equal ">="
decimal integer "0"
equals "="
greater than ">"
decimal integer "7"

***
Test Case 32: Equals
input: 1 = 2 == != 8 =!
expected output:
integer: 1
equals symbol: =
integer: 2
equals symbol: =
equals symbol: =
not equals: !=
int: 8
equals: =
exclamation point: !
actual output:
integer: 1
equals symbol: =
integer: 2
equals symbol: =

equals symbol: =
not equals: !=
int: 8
equals: =
exclamation point: !

Test Case 33: And / Logical And
input: &u & && f& &= =&
expected output:
address: &u
and: &
logical and: &&
word: f
and:&
and/equals: &=
equals: =
and: &
actual output:
bitwise and "&"
word "u"
bitwise and "&"
logical and "&&"
word "f"
bitwise and "&"
bitwise and "&"
equals "="
equals "="
bitwise and "&"

Test Case 34: Or / Logical Or
input: | ghu8 || |= =|
expected output:
or: |
word: ghu
int: 8
logical or: ||
or/equals: |=
equals: =
or: |
actual output:
bitwise or "|"
word "ghu8"
logical or "||"
bitwise or "|"

equals "="
equals "="
bitwise or "|"

Test Case 35: Bitwise Exclusive Or
input: ^ x^5 ^= 0 =^
expected output: bitwise exclusive or: ^
power: x^5
exclusive or equals: ^=
int: 0
equals: =
bitwise exclusive or: ^
actual output:
bitwise exclusive or "^"
word "x"
bitwise exclusive or "^"
decimal integer "5"
bitwise exclusive or "^"
equals "="
decimal integer "0"
equals "="
bitwise exclusive or "^"

Test Case 33: "Other Symbols ~ , ` , | "
input: ',. `~\|
expected output: apostrophe: '
comma: ,
period: .
other symbol: `
other symbol: ~
other symbol: |
actual output: