RAFFY BASAL

PSMDSRC103

Assignment

Perform the following tasks:

1. Type the expressions below, and try to explain what's happening in each case.

```
2 ** 16
# Multiply 16 to the 4th power

2 / 5, 2 / 5.0
# Fraction, Interger and Float tyoe

"spam" + "eggs"
# Combination of two string type

S = "ham"
# Assigning value to a variable

"eggs " + S
# Combination of two string type, S was call out which value is "ham"

S*5
# Repaeats it self 5 times

S[:0]
# Represents the string that assign to "S"
# The left side of the colon, "No Value", indicates that all values
will be shown
# The right side of the colon, "0", indicate that no value will be
shown

"green %s and %s" % ("eggs", S)
# The expression `"green %s and %s" % ("eggs", S)` is using string
formatting in Python.
# The `%s` is a placeholder used for string format, it indicates where
a string value should be inserted when formatting a string.
# % ("eggs", S) is a tuple which represents %s



('x',) [0]
# [0] indicates, accessing the first value of the tupple which is 'x'
```

```
('x', 'y') [1]
# [0] indicates, accessing the second value of the tupple which is
'y'.

L = [1,2,3] + [4,5,6]
# Assigning the value of 'L'
# From concatenation, it combines the two list into one list.

L, L[:], L[:0], L[-2], L[-2:]
# L = Entire List
# L[:] = All values will be shown
# L[:0] = gives an empty list since it slices from the start up to
# L[-2] = Display the second from the last value of the list
# L[-2:]= slices the list from the end as indicated on the index

([1,2,3] + [4,5,6]) [2:4]
# 2 represents where the Index starts and 4 represents the 4th value
where it ends

[L[2], L[3]]
# Accessing elements at indices 2 and 3

L.reverse(); L
# Reversing the element of the list

L.sort(); L
# Sorting the value from the ascending order

L.index(1)
# Returns index 3, returns decending manner

{'a':1, 'b':2} ['b']
# Accessing a dictionary which 'b' has the value of 2

D = {'x':1, 'y':2, 'z':3}
# Creating a dictionary

D['w'] = 0
# adding new key and value in the dictionary

D['x'] + D['w']
# adding to values from the dictionary, '1+0=1'

D[(1,2,3)] = 4
# adding to a dictionary of [(1,2,3)] with the value of 4

D.keys()
# displays the keys used in dictionary

D.values()
# Displays the value asign in each keys
```

```
key_to_check = ((1,2,3))
key_exist = key_to_check in D
print(key_exist)
# check if the value of the key exist in the dictionary.
# I use 'in' because this is supported by the version of python used.

[[]], ["", [], {}, None]
# Nested List. The outer list contains two elements.
# 1. The first element is an empty list `[]`.
# 2. The second element is another list that contains four items:
# - An empty string `""`
# - An empty list `[]`
# - An empty dictionary `{}`
# - A value of `None`
```

2. Define a list named L that contains four strings or numbers. Experiment with the following boundary cases.

Attempt to index out of bounds. (Ex. L[66])

Attempt to slice out of bounds. (Ex. L[-9999:100])

Try extracting a sequence in reverse. Make the lower bound higher than the upper bound (Ex. L[3:1]). Attempt to perform L[3:1] = ['?']. Explain the output.

```
L = [10,20,30,40,50,60,70,80,90,100]

# Attempt to index out of bounds. (Ex. L[66])
L[8]

# Attempt to slice out of bounds. (Ex. L[-9999:100])
L[3:9]

# Try extracting a sequence in reverse.
L.reverse()
print(L)

# Make the lower bound higher than the upper bound (Ex. L[3:1]).
L[3:1]
# Nothing to display, higher bound must have greater-than-value than
the lower bound or it will return a '[]'.
```

3. Define another list L, perform the following and use comments to explain:

Assign a list as one of its elements.

Assign an empty list to a range.

Delete an item in your list using the del statement.

Delete an entire range from your list.

Assign a non-sequence value to a slice.

```
L = []
L.append(L)
print ("Assign a list as one of its elements", L)
# creating a list with empty [], to append the list "L" to be one of
its element

empty_range =[[] for _ in range(10)]
print("Assign an empty list to a range.", empty_range)
# creating an empty [] for a list with a range of 10

L = [5,10,15,20,25,30,35,40,45,50]
del L[1]
print ("List after Deleting an item using the del statement", L)
# removing the number 10, 1st index

del L[-3:]
print ("List after Deleting a range of items", L)
# deleting a range from 40, the 3rd index from the last value of the
list

L = [5,10,15,20,25,30,35,40,45,50]
L [2:4] = [100]
print ("Assign a non-sequence value to a slice.", L)
# assigning new value for index ranges 2 and 3
```

4. Create a dictionary named D with three entries, for keys a, b, and c.

What happens if you try to index a nonexistent key d (D[‘d’])?

What does python do if you try to assign to a nonexistent key?

Compare this to out of bounds assignments.

```
D = {'a': 1, 'b': 2, 'c': 3}
# Attempt to index a nonexistent key 'd'
#print(D['d'])
# Key error, it is not found in the dictionary

# Attempt to assign to a nonexistent key 'd'
D['d'] = 4  # This will create the key 'd' with a value of 4
print (D) # it will give a dictionary that added the key 'd'

L = [1,2,3,4,5]
L.append(5)
print(L)
# print (L[5])
# Cannot assign and add to index 5 because it is out of bound. But you
can append thru elements, the value only not the index itself.
# unlike 'D' that you can add because of the use of {} which is used
to store and retreive data
```

5. Run tests to answer the following questions:

What happens when you try to use the + operator on different/mixed types?

Does + work if one of the operands is a dictionary?

Try list.append() (use an actual list you created). Does the append() method work for both lists and strings?

Try concatenating (+) two lists or strings, what happens?

```
# 1.Different mixed types
mixed = int(20) + float(20.5)
# Possible for int and float
mixed = int(30) + str(50)
# print(mixed)
# not possible for int and str, not supported operand

# 2. Does + work if one of the operands is a dictionary?
```

```
D = {'a':1}
result = D + {'b':2}
# print(result)
# unable to use the '+' operant to an dictionary

# 3. Try list.append() (use an actual list you created). Does the
append() method work for both lists and strings?
L = [1,2,3,4,5]
L.append('Add')
# unssuported to append str in list type of data
# Str has no attribute append

# 4. Try concatenating (+) two lists or strings, what happens?
# Concat Two List
L1 = [1,2,3,4,5]
L2 = [6,7,8,9,10]
concatenated = L1 + L2
print (concatenated)
# Concat Two Strings
S1 = "Hello,"
S2 = " Philippines"
concat = S1 + S2
concat
# Concat strings and list
con = L1 + S1
con
# It is possible to concatenate Lists or String respectively, but not
possible to List and String.
```

6. Define a string S of four characters. Then type the following expression: S[0] [0] [0] [0]. Explain what is happening.

```
S = "DASH"
result = S[0][0][0][0]
result

# it always retreives only the first letter of the string.
# "DASH" is only one length as string, and string do not support
indexing.
```

7. Define a string S of 4 characters again: S = “spam”. Write an assignment that changes the string to “slam”, using only slicing and concatenation. Does it work? Try index assignment.

```
# SPAM to SLAM
S = "spam"
S = S[0] + "c" + S[2:]
S

# Index Assignment
S[1] = "c"
S
# it does work with slicing and concatenate, but id doesn't work with
index assignment
    # str does not support assignment
```