Create Rubric

60 points

List View Grid View

Create your rubric now or come back to it later. You can also make edits to your rubric while grading.

Q1

6 points

Answer the following conceptual questions regarding Python. Unless otherwise stated, assume there is only one correct answer. For the short answer question, please limit your answer to a few sentences. Note that, in order to receive credit for your short answer, you must choose the correct choice first.

Q1.1 Conceptual

1 point

What is the output of the following code?

```
(lambda a: lambda b: lambda c: lambda d: lambda e: lambd
```

Error - Lambda functions need to be stored in a variable to be directly called

Error - There are not enough function calls

0

Lambda function

None of the above

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

Add Rubric Item | Create Group | Import...

Q1.2

1 point

Why are lists referred to as “ordered data structures” whereas dictionaries are referred to as “unordered data structures”?

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

Iteration through a list is possible whereas for a dictionary it is not

Indices in list are directly stored in the list with their elements whereas in a dictionary they are implicitly stored

Indices in lists map to elements whereas there is no such thing in dictionaries

Dictionaries are ordered based on their keys whereas in lists they are ordered based on their indices

None of the above

2 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

Q1.3

2 points

Assume I have a dictionary where the keys are the 26 lowercase letters of the English alphabet and the values are their corresponding indices, both represented as characters:

```
d = {"a": "0", "b": "1", "c": "2", ..., "h": "7", "i": "
```

Is it possible to produce the string “cs88” using only this dictionary?

If so, provide the Python line(s) that, using the dict `d`, assigns the string `"cs88"` to the variable `cs88_str`, eg:

```
# FILL IN YOUR PYTHON CODE HERE
# (Feel free to use more lines. You don't have to use al
__________________
__________________
cs88_str = _________   # FILL ME IN. Should evaluate to "cs88"
```

If not, explain why.

You may use any dictionary methods defined in class; assume that they return lists.

Notably, you may not directly create the string "cs88", eg the following are NOT valid solutions:

```
# INVALID solutions
cs88_str = "cs88"
cs88_str = "cs" + "88"
```

1 +2.0

Fully correct (selected yes + correct explanation)

2 +1.0

Selected yes, explanation is partially correct

3 +0.5

Selected yes, but explanation is incorrect/blank

4 +0.0

Incorrect (select this if "no" is selected even if the explanation is correct)/Blank

+ Add Rubric Item | Create Group | Import...

Yes

No

Your explanation here:

Q1.4

1 point

Suppose we run the following code until it is sequentially finished or it errors:

```
lst = ['C', 8, 8, 'C']
```

Which of the following will mutate the list `lst` above?

[✓] `result = lst`

`result.pop()`

[ ] `result = lst + [4]`

[ ] `result = list(map(lambda x: x*2, lst))`

[✓] `lst.append(4)`

Note: for choice A, it should read as (this is a deficiency of Gradescope's formatting):

```
result = lst
result.pop()
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1	+1.0	2 correct choices selected
2	+1.0	Correct
3	+0.0	Incorrect
4	+0.5	1 correct choice selected
5	+0.0	Blank
6	-0.5	1 incorrect choice selected
7	-1.0	2 incorrect choices selected

+ Add Rubric Item | Create Group | Import...

Q1.5

1 point

Suppose we run the following code until it is sequentially finished or it errors:

```
a = [0, 7, [2, 3, [[2, [0]], 0]], 4]

b = a[2]

c = b[len(b)-1]
c.append(8)

d = c[0]
d.pop(0)
```

What is the output of `print(a)` ?

If any of the lines results in a Python error, choose the option "The code errors".

- `[0, 7, [3, [[2, [0]], 0, 8]], 4]`
- `[0, 7, [3, [[2], 0, 8]], 4]`
- `[0, 7, [2, 3, [[[0]], 0, 8]], 4]`
- `[0, 7, [2, 3, [], 0, 8]], 4]`
- The code errors

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q2 Environment Diagrams

5 points

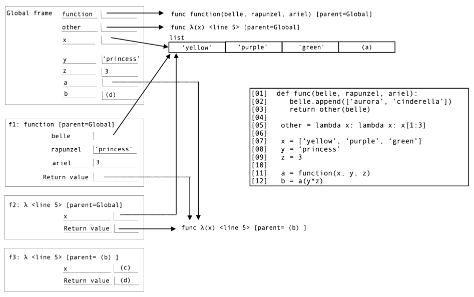
Fill in the blanks to complete the environment diagram. All the code used is in the box to the right, and the code runs to completion.

```
def function(belle, rapunzel, ariel):
    belle.append(['aurora', 'cinderella'])
    return other(belle)

other = lambda x: lambda x: x[1:3]

x = ['yellow', 'purple', 'green']
y = 'princess'
z = 3

a = function(x, y, z)
b = a(y * z)
```


Global frame
function
other
x
y
'princess'
z
3
a
b
(d)
func function(belle, rapunzel, ariel) [parent=Global]
func λ(x) <line 5> [parent=Global]
list
'yellow'
'purple'
'green'
(a)
f1: function [parent=Global]
belle
rapunzel
'princess'
ariel
3
Return value
f2: λ <line 5> [parent=Global]
x
Return value
func λ(x) <line 5> [parent= (b)]
f3: λ <line 5> [parent= (b)]
x
(c)
Return value
(d)

```
[01]  def func(belle, rapunzel, ariel):
[02]     belle.append(['aurora', 'cinderella'])
[03]     return other(belle)
[04]
[05]   other = lambda x: lambda x: x[1:3]
[06]
[07]   x = ['yellow', 'purple', 'green']
[08]   y = 'princess'
[09]   z = 3
[10]
[11]   a = function(x, y, z)
[12]   b = a(y*z)
```

Q2.1

2 points

What is the last element of `x`? (blank a)

- `['cinderella']`
- `['aurora']`
- `['green']`
- `['aurora', 'cinderella']`

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +2.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q2.2

1 point

What is the parent frame of the function that the variable `a` points to? (blank b)

- Global frame
- Frame f1
- Frame f2
- Frame f3

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q2.3

1 point

What is the input value `x` for the lambda function executed in frame f3? (blank c)

'princessprincessprincess'

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct 'princessprincessprincess'

2 +0.0
Incorrect

3 +1.0
Correct

4 +1.0
"princessprincessprincess"

+ Add Rubric Item | Create Group | Import...

Q2.4

1 point

What is the value of the variable `b`? (blank d)

'rin'

'purple', 'green'

'ri'

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q3 What Would Python Print?

6 points

For each expression below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. If an error occurs, write “Error” (if any lines are displayed before the error, include those in your output). If the expression evaluates to a function, write “Function”.

Q3.1

1 point

```
a = [1, 2, 3]
b = a
c = a + b
b[0] = 10
```

```
>>> a
```

[10, 2, 3]

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 **+1.0**
[10, 2, 3]

2 **+1.0**
Correct

3 **+1.0**
[10,2,3]

4 **+0.0**
Incorrect

+ Add Rubric Item | **Create Group** | **Import...**

Q3.2

1 point

```
>>> b
```

[10, 2, 3]

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 **+1.0**
[10, 2, 3]

2 **+0.0**
Incorrect

3 **+1.0**
Correct

4 **+1.0**

[10,2,3]

+ Add Rubric Item | Create Group | Import…

Q3.3

1 point

```
>>> c
```

[1, 2, 3, 1, 2, 3]

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
[1, 2, 3, 1, 2, 3]

2 +0.0
Incorrect

3 +1.0
Correct

4 +1.0
[1,2,3,1,2,3]

+ Add Rubric Item | Create Group | Import…

Q3.4

1 point

Assume that this is a new environment; every variable defined above is no longer accessible

```
a = [1, 2, 3]
b = [4, 5, 6]
c = a.append(b)
```

```
>>> a
```

[1, 2, 3, [4, 5, 6]]

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
[1, 2, 3, [4, 5, 6]]

2 +0.0
Incorrect

3 +1.0
Correct

4 +1.0

[1,2,3,[4,5,6]]

5 +1.0

[1,2,3, [4,5,6]]

+ Add Rubric Item | Create Group | Import...

Q3.5

1 point

```
>>> c
```

None

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

None

2 +0.0

Incorrect

3 +1.0

Correct

4 +1.0

none. "no output" is fine too. a blank submission is also fine.

+ Add Rubric Item | Create Group | Import...

Q3.6

1 point

Assume that this is a new environment; every variable defined above is no longer accessible

```
s = "Hello"
s[0] = 'J'
```

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

Error

2 +1.0

```
>>> s
```

Error

Correct

3 +0.0

Incorrect

4 +1.0

error

+ Add Rubric Item | Create Group | Import...

Q4 Debugging

4 points

You are trying to become the next member of CS88's course staff. As part of your interview, you have received some code and are asked to debug it. In the code below, identify the three bugs and indicate how you would fix them. In your answer, you must fix the bugs in the order they appear.

The below function, `extend()`, takes in two lists and concatenates them, outputting the result as a new list. Furthermore, the two input lists should have all of their elements removed. You may assume that the arguments passed into the function are valid; that is, they themselves will never cause the function to error. You may also assume that `lst1` and `lst2` initially contain at least one element each.

Hint: calling `list()` on an existing list will return a copy of that list.

```
def extend(lst1, lst2):
    """
    >>> lst1 = [1, 2, 3, 4]
    >>> lst2 = [4, 5, 6, 7]
    >>> prod = extend(lst1, lst2)
    >>> prod
    [1, 2, 3, 4, 4, 5, 6, 7]
    >>> lst1
    []
    >>> lst2
    []
    """
    lst = []
    for e in lst1:
        lst = lst.append(e)
        x = lst1.remove(e)
    for e in range(0, len(lst2)):
        lst.append(e)
        x = lst2.pop(e)
    return lst
```

Q4.1

2 points

Identify and fix the first bug.

1	+2.0 Fully correct
2	+1.0 Either only correctly identifies bug or only correctly fixes bug
3	+1.0 Fixes a bug, but there's still a bug remaining in their proposed fix.
4	+0.0 Incorrect/Blank

+ Add Rubric Item | Create Group | Import...

Q4.2

1 point

Identify and fix the second bug.

1 +1.0

Fully correct

2 +0.5

Either only correctly identifies bug or only correctly fixes bug. Or: reasoning is faulty, but fix is correct.

3 +0.5

Fixes a bug, but there's still a bug in their proposed fix.

4 +0.0

Incorrect/Blank

+ Add Rubric Item | Create Group | Import...

Q4.3

1 point

Identify and fix the third bug.

1 +1.0

Fully correct

2 +0.5

Either only correctly identifies bug or only correctly fixes bug

3 +0.5

Fixes a bug, but there is another bug in their proposed fix.

4 +0.0

Incorrect/Blank

+ Add Rubric Item | Create Group | Import...

Q5 Control (loops, if statements, etc.)

5 points

Welcome to the C88C Summer Camp! As a camper, help your fellow campers make bracelets to remember this summer forever by! Implement the function `make_bracelet()` which returns a bracelet where each bead is taken from the `beads` list, repeating the beads from the beads list until the bracelet has reached the specified length. Every nth bead should be a special bead from the `special_beads` list, cycling through `special_beads` as needed. You can assume that LENGTH is greater than or equal to 1, `n` is greater than or equal to 2, there is at least one bead in `beads`, at least one special bead in `special_beads`, and all beads are string values.

```
def make_bracelet(beads, special_beads, length, n):
    >>> beads = ['red', 'blue', 'green']
    >>> special_beads = ['gold', 'silver']
    >>> make_bracelet(beads, special_beads, 10, 3)
    ['red', 'blue', 'gold', 'green', 'red', 'silver', 'blue', 'green', 'gold', 'red']

    bracelet = []
    special_index = 0
    bead_index = 0
    for i in range(1, length + 1):
        ______(a)________:
            bracelet.append(special_beads[_____(b)___________])
            special_index += 1
        ______(c)________:
            bracelet.append(beads[________(d)________])
            bead_index += 1
    return ______(e)___________
```

Q5.1

1 point

Fill in blank (a)

if i % n == 0:

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

`if i % n == 0`

2 +0.0

Incorrect

3 +1.0

Correct

4 +1.0

if i % n == 0:

5 +1.0

if i%n==0

6 +1.0

if i%n==0:

7 +1.0

if i%n == 0:

8 +1.0

if i%n == 0

+ Add Rubric Item | Create Group | Import...

Q5.2

1 point

Fill in blank (b)

`(special_index + 1) % len(special_beads)`

`special_index % len(beads)`

`special_index % len(special_beads) + 1`

`(special_index - 1) % len(special_beads)`

`special_index % len(special_beads)`

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

special_index % len(special_beads)

2 +0.0

Incorrect

3 +1.0

Correct

+ Add Rubric Item | Create Group | Import...

Q5.3

1 point

Select all possible solutions for blank (c)

Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

[✓] `elif i+1%n != 0`

[ ] `elif bead_index < n`

[✓] `else`

[ ] `elif bead_index`

`else:`

2 +0.0
Incorrect

3 +1.0
Correct

+ Add Rubric Item | Create Group | Import...

Q5.4

1 point

Fill in blank (d)

bead_index % len(beads)

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
`bead_index % len(beads)`

2 +0.0
Incorrect

3 +1.0
Correct

4 +1.0
bead_index%len(beads)

+ Add Rubric Item | Create Group | Import...

Q5.5

1 point

Fill in blank (e)

bracelet

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
`bracelet`

2 +0.0

Incorrect

3 +1.0

Correct

+ Add Rubric Item | Create Group | Import...

Q6 Higher Order Functions

8 points

Implement a higher order function `list_dict()` that takes in a string, `s`, and returns a function, `convert`, that takes in either a list or a dictionary, `seq`. If `s` is `"list"`, `seq` will be a list that `convert` should convert into a dictionary, while if `s` is `"dict"`, `seq` will be a dictionary that `convert` should convert into a list.

When converting from a list to a dictionary, the indices should be the keys while the elements should be the values, and vice versa; see the doctests for details.

When converting from a dictionary to a list, assume that the input dict keys are always "valid list indices", eg: the dict keys are integers that start from `0` and strictly increase by 1; see the doctests for details.

To further clarify dict inputs:

```
# VALID dict inputs
{0: "pie", 1: "cake", 2: "bread"}
{0: "turtle"}

# INVALID dict inputs
{2: "meow", 5: "hi"} is INVALID because the keys don't start at 0 and there are nonconsecutive keys
{1: "hi"} is INVALID because the keys don't start at 0
```

You may assume that `s` will always only be either `"list"` or `"dict"`. You may also assume that `seq` will always match the type specified by `s`.

You may not need to use all of the provided lines, but you cannot use more lines than the ones provided. The staff solution uses 9 lines.

Hint: Use an `if` statement to check what `s` is. From there, declare your dictionary or list and fill it in accordingly.

```
def list_dict(s):
    """
    >>> lst = [1, 2, 3]
    >>> c_lst = list_dict("list")
    >>> c_lst(lst)
    {0: 1, 1: 2, 2: 3}
    >>> d = {0: "pie", 1: "cake", 2: "bread"}
    >>> c_dict = list_dict("dict")
    >>> c_dict(d)
    ["pie", "cake", "bread"]
    """
    def convert(seq):
        ______________(1)______________
        ______________(2)______________
        ______________(3)______________
        ______________(4)______________
        ______________(5)______________
        ______________(6)______________
        ______________(7)______________
        ______________(8)______________
        ______________(9)______________
        ______________(10)______________
        ______________(11)______________
        ______________(12)______________
        ______________(13)______________
        ______________(14)______________
        ______________(15)______________
        ______________(16)______________
    return convert
```

Q6.1

8 points

1 **+8.0**

Fully correct

2 **+2.0**

Correctly identifies different cases (i.e. through an `if` statement)

3 **+4.0**

Correctly handles different cases

4 **+2.0**

Returns the values correctly

5 **+0.0**

Incorrect/Blank

+ Add Rubric Item | Create Group | Import...

Q7 Lists

7 points

Q7.1

4 points

IKEA needs help updating their store inventory. Implement the function `add_incoming_shipment()` which takes in:

- `location`: the specific IKEA location's store inventory, represented as a dictionary that maps a `str furniture_category` to a `list` of `str furniture_name`'s.
- `furniture_category`: the category associated with the said specific piece of furniture, represented as a string. Ex: `"Bedroom"`.
- `furniture_name`: the specific piece of furniture you need to sort into a category, represented as a string. Ex: `"HAUGA"`

`add_incoming_shipment()` should **modify** the input `location` dict, either by adding the `furniture_name` to its corresponding `furniture_category` if it already exists in `location`, or adding a new `furniture_category` mapped to the corresponding `furniture_name`.

```
def add_incoming_shipment(location, furniture_category,
    """
    >>> emeryville_loc = {'Bedroom': ['HAUGA', 'TUFJORD'
    >>> add_incoming_shipment(emeryville_loc, 'Kitchen',
    >>> emeryville_loc
    {'Bedroom': ['HAUGA', 'TUFJORD'] , 'Kitchen': ['POKA
    >>> add_incoming_shipment(emeryville_loc, 'Bathroom'
    >>> emeryville_loc
    {'Bedroom': ['HAUGA', 'TUFJORD'] , 'Kitchen': ['POKA
    """
    # FILL ME IN!
```

1 **+4.0**

```
if furniture_category in location: #can use location
       location[furniture_category].append(furniture_
else:
        location[furniture_category] = [furniture_nam
```

2 **+4.0**

```
if furniture_category in location: #can use location
       location[furniture_category] += [furniture_nam
else:
       location[furniture_category] = [furniture_name
```

3 **+4.0**

```
if furniture_category not in location:
      location[furniture_category] = []
location[furniture_category].append(furniture_name)
```

4 Case 1: Adding the furniture_name to its corresponding furniture_category if it already exists in location

5 Case 2: Adding a new furniture_category mapped to the corresponding furniture_name

6 **-0.2**

has a return statement

7 **-0.2**

Minor syntax error but otherwise correct/mostly correct solution

8 **-0.5**

Conceptual error, but otherwise correct/mostly correct solution

9 **+0.0**

Blank/Incorrect

+ Add Rubric Item | Create Group | Import...

Q7.2

3 points

Uh oh! Some of the IKEA locations are getting their shipments all jumbled up! Implement the function `sort_shipments` which takes in:

- `jumbled_shipment`: a `list` of furniture pieces represented as 3-element lists with the 0th index corresponding to its "tag" or the name of the location (`string`) the furniture piece should have gone to, the 1st index corresponding to the name of the furniture piece, and the 2nd index corresponding to the category the furniture piece belongs to.
- `loc1:` the first location whose furniture pieces were mixed into the jumbled shipment
- `loc2`: the second location whose furniture pieces were mixed into the jumbled shipment

and returns two lists, where the first list contains all the furniture pieces for `loc1` and the second list contains all the furniture pieces for `loc2`. For each returned list, make sure to only return the `[str furniture_name, str furniture_category]` (eg location is omitted): see the doctests for details.

Note: If an entry in `jumbled_shipment` doesn't correspond to either `loc1` or `loc2`, you can omit this entry from the output:

```
# Entry for "loc_missing" is omitted from outputs
>>> emeryville_loc, sf_loc = sort_shipments([["loc_miss
>>> emeryville_loc
[['TUFJORD', 'Bedroom']]
>>> sf_loc
[['POKAL', 'Kitchen']]
```

You may not need to use all of the provided lines, but you cannot use more lines than the ones provided. The staff solution uses 5 lines.

1 **+3.0**

```
loc = item.pop(0)
if loc == loc1:
        loc1_lst.append(item)
    else:
        loc2_lst.append(item)
```

2 **+3.0**

```
if item[0] == loc1:
    loc1_lst.append(item[1:])
elif item[0] == loc2:
    loc2_lst.append(item[1:])
```

3 **+3.0**

```
if item[0] ==loc1:
    loc1_lst.append ([item[1],item[2]])
elif item[0] ==loc2:
    loc2_lst.append([item[1], item[2]])
```

4 Correctly compares the name of the location of a furniture piece to loc1 and loc2

5 Correctly adds furniture pieces to loc1_lst and loc 2_lst

6 **+1.0**

Correctly removes the name of the location corresponding the final output lists: `item[1:]`, `item.pop(0) #be careful to ensure student pops off O` , `[item[1], item[2]]`, etc.

7 **+0.25**

Attempts to remove the name of the location corresponding to each furniture piece from the final

```
def sort_shipments(jumbled_shipment, loc1, loc2):
    """
    >>> mixed_shipment = [['Emeryville', 'TUFJORD', 'Bed
    ['Emeryville', 'TISKEN', 'Bathroom'], ['San Francisc
    >>> emeryville_loc, sf_loc = sort_shipments(mixed_sh
    >>> emeryville_loc
    [['TUFJORD', 'Bedroom'], ['TISKEN', 'Bathroom'], ['H
    >>> sf_loc
    [['POKAL', 'Kitchen'], ['FROSLOV', 'Living Room'], [
    """
    loc1_lst = []
    loc2_lst = []
    for item in jumbled_shipment:
        ______________(1)______________
        ______________(2)______________
        ______________(3)______________
        ______________(4)______________
        ______________(5)______________
        ______________(6)______________
        ______________(7)______________
        ______________(8)______________
        ______________(9)______________
        ______________(10)______________
    return loc1_lst, loc2_lst
```

output lists

8 **-0.2**
Minor syntax error, but otherwise correct solution

9 **-0.5**
Conceptual error, but otherwise correct solution

0 **+0.0**
Blank/Incorrect

+ Add Rubric Item | Create Group | Import...

Q8 Recursion

7 points

Write a recursive function `recursive_sum()` that takes in a nested list, `lst`, and returns the sum of all of its non-list elements. For example, consider the following doctest:

```
>>> lst = [
    1,
    [4, 0],
    [
        3,
        [2, 7],
        [10]
    ],
    [[2]]
]
>>> recursive_sum(lst)
29
>>> lst2 = []
>>> recursive_sum(lst2)
0
```

The input lists will generally follow the format above. You may assume that `lst` will always be a list, and you may also assume that the only non-list elements will be integers.

Note: to check whether an element is a list,
you can use the `isinstance` function like so:

```
>>> lst = [1, 2, 3]
>>> isinstance(lst, list) # checks if lst is a list object
True
>>> num = 1
>>> isinstance(num, list) # checks if num is a list object
False
```

Enter your solution below:

```
def recursive_sum(lst):
    if _______(a)_______:
        return 0
    if _______(b)_______:
        _______(c)_______
    return _______(d)_______
```

Q8.1

1 point

What goes in blank (a)?

len(lst) == 0

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

Correct len(lst) == 0:

2 +1.0

Correct

3 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

Q8.2

2 points

What goes in blank (b)?

isinstance(lst[0], list)

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +2.0

Correct isinstance(lst[0], list), OR `not isinstance(lst[0], list)` if the other blanks align correctly

2 **+0.0**

Incorrect

3 **+2.0**

Correct

+ Add Rubric Item | Create Group | Import...

Q8.3

2 points

What goes in blank (c)?

return recursive_sum(lst[0]) + recursive_sum(lst[1:])

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 **+2.0**

Correct: `return recursive_sum(lst[0]) + recursive_sum(lst[1:]` . OR: `return lst[0] + recursive(sum(lst[1:]))` if Q8.2 agrees

2 **+2.0**

Correct

3 **+1.0**

Partially correct, eg `summ = lst[0] + recursive_sum(lst[1:])`, `return recursive_sum(lst[0])`, `sum += recursive_sum(lst[0])`

4 **+0.0**

Incorrect

+ Add Rubric Item | Create Group | Import...

Q8.4

2 points

What goes in blank (d)?

lst[0] + recursive_sum(lst[1:])

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +2.0
Correct, `return lst[0] + recursive_sum(lst[1:])`

2 +2.0
Correct

3 +1.0
Partially correct

4 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q9 Object Oriented Programming

6 points

Provided is an implementation of the `Person` and `Musician` classes that we will be working with:

```
class Person:
    def __init__(self, age):
        self.age = age
        self.energy = 10

    def have_birthday(self):
        old_age = self.age
        self.age += 1
        print(f"Happy birthday to me: {old_age} -> {self.age} age")
        return self.age

    def have_fun(self):
        old_energy = self.energy
        self.energy -= 1
        print(f"That was fun: {old_energy} -> {self.energy} energy")
        return self.energy

class Musician(Person):
    def __init__(self, age, instrument):
        super().__init__(age)
        self.instrument = instrument

    def have_fun(self):
        # (Question 1) FILL ME IN

    def practice(self, num_hours):
        self.energy -= num_hours
        return self.energy
```

Q9.1

3 points

Finish the Musician `have_fun()` method, which should behave as follows:

- Musician.have_fun() behaves EXACTLY LIKE the parent class's `have_fun()` (aka "Musician has fun exactly like Person"), but loses an additional energy.
- `have_fun()` should return the new energy

```
>>> paul = Musician(82, "bass")
>>> paul.energy
10
>>> paul.have_fun()
That was fun: 10 -> 9 energy
8
>>> paul.energy
8
```

1 **+3.0**
Correct

2 **+2.5**
Very close. Calls super(), but with one minor bug. Ex: incorrectly passes in self to have_fun(), eg: `super().have_fun(self)`, or missing the return stmt

3 **+2.0**
Correct behavior, but does not call super()

4 **+1.5**
Incorrect behavior, but close-ish: either doesn't set energy correctly, or the print stmt is incorrect (eg prints -2 instead of -1)

5 **+1.0**
Wrong, but is somewhat on the right path: uses class variables instead of instance variables (eg Person.energy vs self.energy), or is missing the print stmt

6 **+0.0**
Incorrect. Notably, does not try to update energy, or is not on the right path at all

+ Add Rubric Item | Create Group | Import...

Q9.2

1 point

Suppose we wanted to keep track of the total number of practice hours across all `Musician`s in the following way:

```
>>> paul = Musician(82, "bass")
>>> ringo = Musician(84, "drums")
>>> Musician.total_practice_hours
0
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 **+1.0**
Correct

```
>>> paul.practice(4)
6
>>> ringo.practice(6)
8
>>> Musician.total_practice_hours
10
```

In the `Musician` class, which choice is the correct way to add the `total_practice_hours` variable?

```
class Musician(Person):
    """
    [Choice A]
    total_practice_hours = 0
    """
    """
    [Choice B]
    self.total_practice_hours = 0
    """

    def __init__(self, age, instrument):
        super().__init__(age)
        """
        [Choice C]
        total_practice_hours = 0
        """
        self.instrument = instrument
        """
        [Choice D]
        self.total_practice_hours = 0
        """
```

Choice A

Choice B

Choice C

Choice D

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

Q9.3

1 point

In the `Musician` class's `practice()` method (reproduced below), which choice is the correct way to update the `total_practice_hours` variable?

```
    def practice(self, num_hours):
        self.energy -= num_hours
        """
        [Choice A]
        self.total_practice_hours += num_hours
        """
        """
        [Choice B]
        total_practice_hours += num_hours
        """
        """
        [Choice C]
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

```
        Musician.total_practice_hours += num_hours
        """
        """
        [Choice D]
        Person.total_practice_hours += num_hours
        """
        return self.energy
```

Choice A

Choice B

Choice C

Choice D

Q9.4

1 point

We'd like to implement a new `Person` subclass, `OddPerson`, that is EXACTLY like a `Person`, but with the following changes:

- In `OddPerson`'s `have_birthday()` method, their age does not increase (aka an `OddPerson` never gets older).
- `have_birthday()` should return their current (unchanged) age

```
>>> odd_keanu = OddPerson(59)
>>> odd_keanu.have_birthday()
59
>>> odd_keanu.age
59
```

For each of the following implementations, mark if it is a correct implementation (choose ALL that apply):

```
# [Choice A]
class OddPerson(Person):
   def have_birthday(self):
       return self.age

# [Choice B]
class OddPerson(Person):
   def have_birthday():
       return OddPerson.age

# [Choice C]
class OddPerson(Person):
    def have_birthday():
        return self.age
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

Correct

2 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

[✓ Choice A]

Choice B

Choice C

Q10 Abstract Data Types

6 points

Consider this Phone abstract data type (ADT), internally implemented as a list:

```
# BEGIN Phone ADT

# Constructors
def make_phone(style, color):
    return [style, color]

# Selectors
def get_style(phone):
    return phone[0]

def get_color(phone):
    return phone[1]

# END Phone ADT
```

Q10.1

2 points

We would like to implement a function `is_same_phone_colors()` that, given a list of pairs of phones, for each pair returns `True` if the paired phones have the same color:

```
>>> phone_a = make_phone("iphone", "pink")
>>> phone_b = make_phone("pixel", "gold")
>>> phone_c = make_phone("iphone", "grey")
>>> phone_d = make_phone("pixel", "grey")
>>> is_same_phone_colors([(phone_a, phone_b), (phone_c,
[False, True]
```

Notably, `is_same_phone_colors()` is a USER of the Phone ADT (eg an "operation", and NOT part of the core Phone ADT itself).

Take a look at the following implementations of `is_same_phone_colors()`, and answer their questions:

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +2.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...

```
def is_same_phone_colors(pairs_of_phones):
    output = []
    for ind in range(len(pairs_of_phones)):
        phone_a = pairs_of_phones[ind][0]
        phone_b = pairs_of_phones[ind][1]
        output.append(get_color(phone_a) == get_color(phone_b))
    return output
```

This implementation produces the desired output AND respects the Phone ADT

This implementation produces the desired output but does NOT respect the Phone ADT

This implementation does not produce the desired output.

This implementation produces an error.

Q10.2

1 point

```
def is_same_phone_colors(pairs_of_phones):
    output = []
    for pair in pairs_of_phones:
        phone_a = get_style(pair)
        phone_b = get_color(pair)
        output.append(get_color(phone_a) == get_color(ph
    return output
```

This implementation produces the desired output AND respects the Phone ADT

This implementation produces the desired output but does NOT respect the Phone ADT

This implementation does not produce the desired output.

This implementation produces an error.

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

Correct

2 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

Q10.3

1 point

Suppose I defined a `phone_to_str()` function that returns a human-friendly string representation of a Phone:

```
>>> phone_a = make_phone("android", "blue")
>>> phone_to_str(phone_a)
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

```
'Phone(style=android,color=blue)'
```

Correct

Notably, `phone_to_str()` is a USER of the Phone ADT (eg an "operation"), and is not part of the core Phone ADT itself.

2 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

Consider this `phone_to_str()` implementation that does produce the desired output (as described above):

```
def phone_to_str(phone):
    str_out = "Phone("
    ind = 0
    while ind < len(phone):
        field_val = phone[ind]
        if ind == 0:
            str_out += f"style={field_val},"
        elif ind == 1:
            str_out += f"color={field_val}"
        ind += 1
    return str_out + ")"
```

Does this function respect the Phone ADT?

Yes

No

Q10.4

1 point

Suppose we want to add a new constructor to the Phone ADT, that allows users to create a Phone without specifying a color, and the default color ("dc") is `"default_color"`:

```
>>> phone_default_color = make_phone_dc("iphone")
>>> phone_blue = make_phone("iphone", "blue")
>>> get_color(phone_default_color)
'default_color'
>>> get_color(phone_blue)
'blue'
```

Notably, this new constructor is part of the Phone ADT.

Consider the following implementations of the `make_phone_dc()` constructor, and answer their questions:

```
def make_phone_dc(style):
    return [style, "default_color"]
```

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0

Correct

2 +0.0

Incorrect

+ Add Rubric Item | Create Group | Import...

This implementation produces the desired output AND respects the Phone ADT

This implementation produces the desired output but does NOT respect the Phone ADT

This implementation does not produce the desired output.

This implementation produces an error.

Q10.5

1 point

```
def make_phone_dc(style):
    return make_phone(style, "default_color")
```

This implementation produces the desired output AND respects the Phone ADT

This implementation produces the desired output but does NOT respect the Phone ADT

This implementation does not produce the desired output.

This implementation produces an error.

⚠ Removing the **Correct** and **Incorrect** rubric items will interfere with auto-grading for this question.

1 +1.0
Correct

2 +0.0
Incorrect

+ Add Rubric Item | Create Group | Import...