

## A LEARNING CONTEXTS AND TASKS

This appendix provides additional detail on the tasks associated with each study.

### A.1 Pretest

The optional pretest, which can be run as timed or untimed, is designed to assess the student's level of knowledge of the subject matter.

<p>Q-1: What is the output from the program below?</p> <pre>def check_systolic(num1):     if num1 &lt; 120:         return 0     elif num1 &lt; 140:         return 1     elif num1 &lt; 180:         return 2     else:         return 3  def check_diastolic(num2):     if num2 &lt; 80:         return 0     elif num2 &lt; 90:         return 1     elif num2 &lt; 110:         return 2     else:         return 3  syst = 135 dias = 100 if check_systolic(syst) == 0 and check_diastolic(dias) == 0:     print ("Normal") elif check_systolic(syst) == 3 or check_diastolic(dias) == 3:     print ("Hypertensive Crisis") elif check_systolic(syst) == 1:     if check_diastolic(dias) &gt; 1:         print ("High Blood Pressure A")     else:         print ("Prehypertension")</pre> <p><input type="radio"/> A. Prehypertension <input type="radio"/> B. High Blood Pressure A <input type="radio"/> C. Hypertensive Crisis <input type="radio"/> D. Normal <input type="radio"/> E. I don't know</p>	<p>Q-1: What will the following code print?</p> <pre>score = 93 if score &gt;= 90:     grade = "A" if score &gt;= 80:     grade = "B" if score &gt;= 70:     grade = "C" if score &gt;= 60:     grade = "D" if score &lt; 60:     grade = "E" print(grade)</pre> <p><input type="radio"/> A. A <input type="radio"/> B. D <input type="radio"/> C. E <input type="radio"/> D. A, B, C, D <input type="radio"/> E. I don't know</p>
	<p>Q-1: What is returned by the following function?</p> <pre>def list_within_list():     alist = [3, [6 7], "cat", [56, 57, "dog"], [ ], 3.14, False]     return alist[2][0]</pre> <p><input type="radio"/> A. An error because of alist[2][0] <input type="radio"/> B. 56 <input type="radio"/> C. 6 <input type="radio"/> D. c <input type="radio"/> E. I don't know</p>
<p>Q-1: What does the following expression output?</p> <pre>print (type(1.0 == 6))</pre> <p><input type="radio"/> A. &lt;class 'bool'&gt; <input type="radio"/> B. &lt;class 'float'&gt; <input type="radio"/> C. &lt;class 'int'&gt; <input type="radio"/> D. False <input type="radio"/> E. I don't know</p>	<p>Q-1: What will the following code print?</p> <pre>def mystery(num_list):     sum = 0     for i in range(0, len(num_list), 2):         num = num_list[i]         sum += num     return sum  list1 = [1, 2, 3, 4, 5] print(mystery(list1))</pre> <p><input type="radio"/> A. 1 <input type="radio"/> B. 9 <input type="radio"/> C. 15 <input type="radio"/> D. None <input type="radio"/> E. I don't know</p>

Figure 26: First 5 questions from 10-question multiple-choice pretest

<p>Q-1: What does the following code output?</p> <pre>def abbrev(first_name, last_name):     print(first_name[0:1] + ". " + last_name.lower())  abbrev("Joanne", "Weathers")</pre> <p> <input type="radio"/> A. J. Weathers  <input type="radio"/> B. Jo. Weathers  <input type="radio"/> C. oa. Weathers  <input type="radio"/> D. J. weathers  <input type="radio"/> E. I don't know         </p>	<p>Q-1: What will the following code print?</p> <pre>def mystery(str):     out = ""     for char in str:         if char == "i":             break         if char == 'a':             continue         out += char     return out  print(mystery("walking"))</pre> <p> <input type="radio"/> A. walking  <input type="radio"/> B. wlking  <input type="radio"/> C. wlk  <input type="radio"/> D. wlkng  <input type="radio"/> E. I don't know         </p>
<p>Q-1: What will the following code print?</p> <pre>def mystery(num_list):     sum = 0     for num in num_list:         if num % 2 == 1:             sum += num     return sum  list1 = [2, 3, 4, 5, 7] print(mystery(list1))</pre> <p> <input type="radio"/> A. 6  <input type="radio"/> B. 8  <input type="radio"/> C. 15  <input type="radio"/> D. 21  <input type="radio"/> E. I don't know         </p>	<p>Q-1: What does the following code print?</p> <pre>game = 'Lost Vikings' print(game[-6:-1])</pre> <p> <input type="radio"/> A. Vikings  <input type="radio"/> B. Viking  <input type="radio"/> C. ikings  <input type="radio"/> D. iking  <input type="radio"/> E. I don't know         </p>
<p>Q-1: What does the following code print?</p> <pre>output = "" x = -5 while x &lt; 0:     x = x + 1     output = output + str(x) + " " print(output)</pre> <p> <input type="radio"/> A. 5 4 3 2 1  <input type="radio"/> B. -4 -3 -2 -1 0  <input type="radio"/> C. -5 -4 -3 -2 -1  <input type="radio"/> D. -5 -4 -3 -2 -1 0  <input type="radio"/> E. I don't know         </p>	

Figure 27: Last 6 questions from 10-question multiple-choice pretest

# Pre Test

Please try to solve each of the following problems to the best of your ability. It is **OK** to not know the correct answers! If you don't know the answer just select option E (I don't know).

## Problems

Time Remaining 20:00

Start

## Feedback

Q-11: Please provide feedback here. Please share any comments, problems, or suggestions.

Write your answer here

Save

Instructor's Feedback

You have not answered this question yet.

Activity: 2 shortanswer (p3-pre-sa)

## Thank You 🙌

🎉 We appreciate your participation in our study.

Figure 28: Pretest final feedback question

## A.2 Introduction to Problem Types

All of the studies included a primer on how to use the interface, which also served to introduce the problem types and help students familiarise themselves with Parson's problems.

Try to solve the following mixed-up code problem. This problem doesn't require any indentation.

Drag the blocks from the left and put them in the correct order on the right. The text in each block defines the order.

*Drag from here*

*Drop blocks here*

1 First block

2 Second block

3 Third block

Check Reset Help me

Parsons (intro-simple-parsons-no-indent-ps)

Figure 29: Introduction to problem types - Students familiarise themselves with the drag and drop interface for Parsons problems

Try to solve the following mixed-up code problem. This problem requires indentation.

Drag the blocks from the left and put them in the correct order on the right with the correct indentation. The text in each block defines the order and indentation.

*Drag from here*

*Drop blocks here*

1 First block

2 Second block

3 Third block that needs to be indented

Check Reset Help me

Parsons (intro-simple-parsons-indent-ps)

Figure 30: Introduction to problem types - Students are introduced to indentation in Parsons problems

Try to solve the following mixed-up code problem. This problem requires indentation and has extra blocks that are not needed in a correct solution.

Drag the blocks from the left and put them in the correct order on the right with the correct indentation. There is an extra block that is not needed in the correct solution.

*Drag from here*

1 Third block that needs to be indented

2a Extra block that is not needed

2b Second block

3 First block

*Drop blocks here*

Check

Reset

Help me

Parsons (intro-simple-parsons-indent-with-dist-ps)

**Figure 31: Introduction to problem types - Students practice parsons problems with distractor blocks i.e. blocks which are not needed in a correct solution**

Finish writing the code for the following problem.

Write a function called `triple(num)` that takes a number `num` and returns the number times 3. For example, `triple(2)` should return 6 and `triple(-1)` should return -3. Look below the code to check for any compiler errors or the results from the test cases. Be sure to `return` the result.

Save & Run

Original - 1 of 1

Show CodeLens

Share Code

```

1 def triple(num):
2     # write code here
3
4 print(triple(2))
5 print(triple(-1))
6
7

```

Activity: 7 ActiveCode (intro-sample-write-code-triple-ps)

**Figure 32: Introduction to problem types - Students practice writing code with unit tests**

### A.3 *python-swap*

Students are tasked with swapping the value of one variable with the value of another variable.

The following has the correct code to 'swap' the values in x and y (so that x ends up with y's initial value and y ends up with x's initial value), but the code is mixed up and contains one extra block which is not needed in a correct solution. Drag the needed blocks from the left into the correct order on the right. Check your solution by clicking on the Check button. You will be told if any of the blocks are in the wrong order or if you need to remove one or more blocks. After three incorrect attempts you will be able to use the Help Me button to make the problem easier.

Drag from here

Drop blocks here

```
1 | # set temp to the value of x
2 | # set y to the value of temp
3 | # initialize the variables
4 | # set y to the value of x
5 | # set x to the value of y
```

Check Reset Help me

Parsons (ps\_swap\_comments\_pp)

Figure 33: *python-swap* - Students reorganise comment blocks which contain the logic of the algorithm for swapping the values of two variables

The following has the correct code to 'swap' the values in x and y (so that x ends up with y's initial value and y ends up with x's initial value), but the code is mixed up and contains one extra block which is not needed in a correct solution. Drag the needed blocks from the left into the correct order on the right. Check your solution by clicking on the Check button. You will be told if any of the blocks are in the wrong order or if you need to remove one or more blocks. After three incorrect attempts you will be able to use the Help Me button to make the problem easier.

*Drag from here*

1    `# set y to the value of temp`  
`y = temp`

2    `# set y to the value of x`  
`y = x`

3    `# set temp to the value of x`  
`temp = x`

4    `# set x to the value of y`  
`x = y`

5    `# initialize the variables`  
`x = 3`  
`y = 5`  
`temp = 0`

*Drop blocks here*

Check
Reset
Help me

Parsons (ps\_swap\_code\_and\_comments\_pp)

Figure 34: *python-swap* - Students practice combining the code with comments blocks to teach the logic of the process

The following has the correct code to 'swap' the values in x and y (so that x ends up with y's initial value and y ends up with x's initial value), but the code is mixed up and contains one extra block which is not needed in a correct solution. Drag the needed blocks from the left into the correct order on the right. Check your solution by clicking on the Check button. You will be told if any of the blocks are in the wrong order or if you need to remove one or more blocks. After three incorrect attempts you will be able to use the Help Me button to make the problem easier.

*Drag from here*

1    `y = temp`

2    `x = 3`  
`y = 5`  
`temp = 0`

3    `y = x`

4    `x = y`

5    `temp = x`

*Drop blocks here*

Check
Reset
Help me

Parsons (ps\_swap\_code\_only\_pp)

Figure 35: *python-swap* - Students are tasked with a Parsons problem to swap the values of two variables

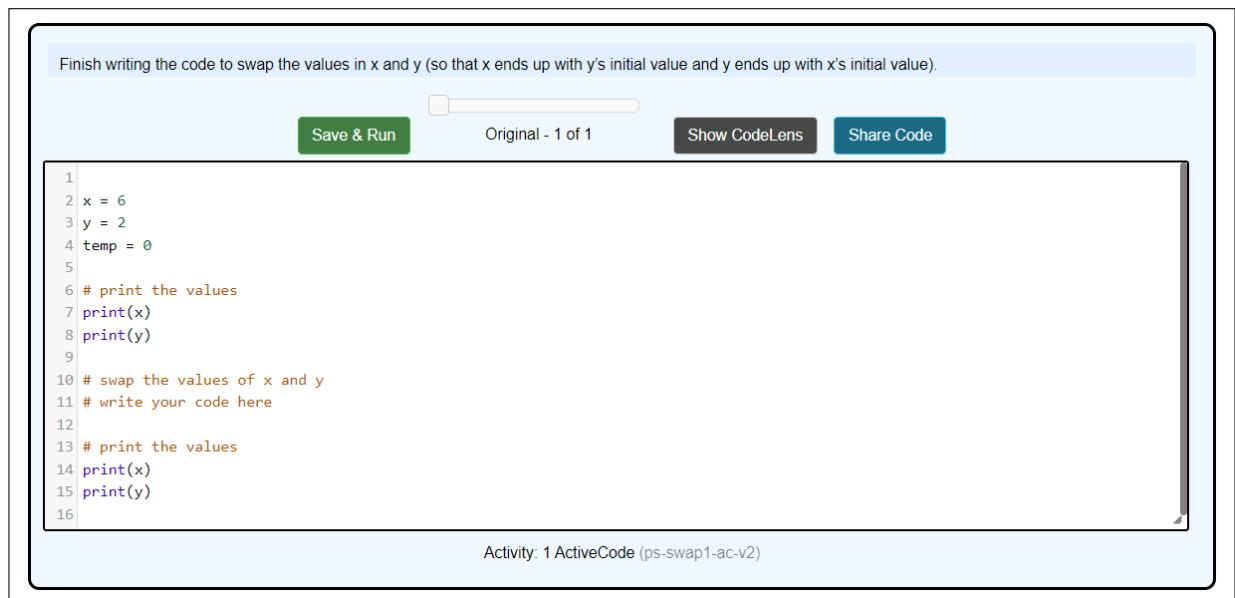


Figure 36: *python-swap* - After completing the Parson's problem exercise, students proceed to write a solution using code



Figure 37: *python-swap* - A similar code-writing task with different variable names for near transfer



#### A.4 p3pt

Students are tasked with defining functions that solve defined problems in Python.

Create the function `get_middle(str)` to return the middle characters from the passed string `str`. If `str` has less than 3 characters then return `str`. If `str` has an odd length then return the middle character. If `str` has an even length return the two middle characters. For example, `get_middle('abc')` returns `'b'` and `get_middle('abcd')` returns `'bc'`.

Drag from here

Drop blocks here

1 | `return str[mid]`

2a | `return str[mid-1:mid+1]`

2b | `return str[mid-1:mid]`

3 | `def get_middle(str):`

4a | `num_chars = len(str)`  
`mid = num_chars / 2`

4b | `num_chars = len(str)`  
`mid = num_chars // 2`

5 | `if num_chars < 3:`

6a | `elif num_chars % 2 == 1:`

6b | `elif num_chars % 2 == 1:`

7 | `else:`

8 | `return str`

Check Reset Help me

Parsons (get-middle-Parsons-Version-pilot)

Figure 38: p3pt - Defining a function which extracts the middle characters from a string (Parsons)

Create the function `combine(names, ages)` that takes in two lists, `names` and `ages` and returns a list of strings in the format `"Name: name, age: age"`. For example, `combine(["Claire", "Jennifer"], [23, 19])` would return `["Name: Claire, age: 23", "Name: Jennifer, age: 19"]`.

Drag from here

Drop blocks here

1a `s="Name: "+names[i]+"", "+age: "+str(ages[i])`

1b `s="Name: "+names[i]+"", "+age: "+ages[i]`

2 `newlist.append(s)`

3 `def combine(names,ages):`

4 `return newlist`

5 `newlist=[]`

6a `for i in range(len(names)):`

6b `for i in range(len(names)-1):`

Check Reset Help me

Parsons (list\_loop\_two\_lists\_pp)

Figure 39: *p3pt* - Defining a function that concatenates two lists according to requirements (Parsons)

Create the function `has22(nums)` below to return `True` if there are at least two items in the list `nums` that are adjacent and both equal to `2`, otherwise return `False`. For example, return `True` for `has22([1, 2, 2])` since there are two adjacent items equal to `2` (at index 1 and 2) and `False` for `has22([2, 1, 2])` since the `2`'s are not adjacent.

Drag from here

Drop blocks here

1a for i in range(len(nums) - 1):

1b for i in range(len(nums)):

2a if nums[i] == nums[i + 1]:

2b if nums[i] == 2 and nums[i + 1] == 2:

3 return False

4 def has22(nums):

5 return True

Check Reset Help me

Parsons (has22\_Parsons-Version-A)

Figure 40: *p3pt* - Defining a function to search a list for adjacent values of two (Parsons)

Create the function `sum13(nums)` to return the sum of the numbers in the list `nums`, returning `0` for an empty list. Except the number 13 is very unlucky, so it does not count and a number that comes immediately after a 13 also does not count. For example, `sum13([13,1,2])` returns `2` and `sum13([1,13])` returns `1`.

Drag from here

Drop blocks here

1

else:

2

found\_13 = True

3

def sum\_13(nums):

4

for num in nums:

5a

return total

or

5b

return Total

6

if found\_13:

7

found\_13 = False

8a

total = 0

found\_13 = True

or

8b

total = 0

found\_13 = False

9a

elif num == 13:

or

9b

elif num = 13:

10

total += num

Check

Reset

Help me

Parsons (sum13-Parsons-version-pilot)

Figure 41: *p3pt* - Defining a function to sum a list of numbers excluding any element that follows the value of 13 (Parsons)



Write a function `combine(names, ages)` that takes in two lists, `names` and `ages` and returns a list of strings in the format `"Name: name, age: age"`. For example, `combine(["Claire", "Jennifer"], [23, 19])` would return `["Name: Claire, age: 23", "Name: Jennifer, age: 19"]`.

☐

[Run](#) Original - 1 of 1 [Show CodeLens](#) [Share Code](#)

```
1 def combine(names, ages):
2
3
```

Activity: 1 ActiveCode (list\_loop\_two\_lists\_ac)

Figure 43: *p3pt* - Defining a function that combines two lists according to requirements (code writing)

Finish the function `has22(nums)` below to return `True` if there are at least two items in the list `nums` that are adjacent and both equal to `2`, otherwise return `False`. For example, return `True` for `has22([1, 2, 2])` since there are two adjacent items equal to `2` (at index 1 and 2) and `False` for `has22([2, 1, 2])` since the `2`'s are not adjacent.

☐

[Run](#) Original - 1 of 1 [Share Code](#)

```
1 def has22(nums):  
2  
3
```

Activity: 1 ActiveCode (has22\_Write)

Figure 44: *p3pt* - Defining a function to search a list for adjacent values of two (code writing)

Finish the function `sum13(nums)` to return the sum of the numbers in the list `nums`, returning `0` for an empty list. Except the number 13 is very unlucky, so it does not count and a number that comes immediately after a 13 also does not count. For example, `sum13([13,1,2])` returns `2` and `sum13([1,13])` returns `1`.

☐

[Run](#) Original - 1 of 1 [Share Code](#)

```
1 def sum13(nums):  
2  
3
```

Activity: 1 ActiveCode (sum13\_writecode\_test\_1\_v2)

Figure 45: *p3pt* - Defining a function to sum a list of numbers excluding any element that follows the value of 13 (code writing)





Write a function `is_descending(nums)` that returns `True` if the numbers in the list `nums` are sorted in descending order and `False` otherwise. If the list `nums` has less than two numbers in it return `True`. For example, `is_descending([2,3,4])` should return `False`, `is_descending([1])` should return `True`, and `is_descending([4,3,2])` should return `True`.

☐

[Run](#) Original - 1 of 1 [Share Code](#)

```
1 def is_descending(nums):
2     #write your code here
3
4
5
6 print(is_descending([2, 3, 4]))
7 print(is_descending([3]))
8 print(is_descending([4, 3, 2]))
9
```

Activity: 1 ActiveCode (pretest\_is\_ascending\_ac-post)

Figure 47: *p3pt* - Posttest based on determining whether a list is in descending order

Fix the `sum67` function below that takes a list of numbers and returns the total of the numbers in the list except for all the numbers whose position is between a 6 and 7 in the list (inclusive). For example, `sum67([1,2])` should return 3 and `sum67([2, 6, 8, 7, 2])` should return 4.

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 def sum67(nums):
2     total = 0           # initialize the total
3     found_6 = True      # initialize a Boolean flag
4     for num in nums:    # loop through the items in a list
5         if found_6 && num == 7:
6             found_6 = False # set the Boolean flag to false
7         elif num == 6:
8             found_6 = True  # set the Boolean flag to True
9         elif found_6:
10            continue       # go back to the top of the loop
11        else:
12            total += num    # add num to total
13        return total      # return the total
14
```

Activity: 1 ActiveCode (sum67\_fix)

Figure 48: *p3pt* - Posttest based on conditional summation

Write a function `olympic(cities, years)` that takes in two lists, `cities` and `years` and returns a list of strings in the format `"City: city, year: year"` . For example, `olympic(["Paris", "London"], [2024, 2012])` would return `["City: Paris, year: 2024", "City: London, year: 2012"]` .

☐

[Run](#) Original - 1 of 1 [Show CodeLens](#) [Share Code](#)

```
1 def olympic(cities, years):
2
3
```

Activity: 1 ActiveCode (lst\_two\_loop\_post)

Figure 49: *p3pt* - Posttest based on combining two lists

## A.5 Introduction to Classes

The *classexp* and *classtog* studies involved object-orientated concepts and programming constructs. Brief instruction on how to define classes are provided as part of these respective studies.

Run the following code

Save & Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 class Book:
2     """ Represents a book object """
3
4     # initializes the values in a new object called self
5     def __init__(self, title, author):
6         self.title = title # set title in self to the passed title
7         self.author = author # set author in self to the passed author
8
9     # returns a string with information about the object self
10    def __str__(self):
11        return "title: " + self.title + " author: " + self.author
12
13 def main():
14     # calls the __init__ method
15     b2 = Book("A Wrinkle in Time", "M. L'Engle")
16
17     # calls the __str__ method
18     print(b2)
19
20     # calls the __init__ method
21     b1 = Book("Goodnight Moon", "Margaret Wise Brown")
22
23     # calls the __str__ method
24     print(b1)
25
26 main()
27
```

Activity: 1 A class to represent a book (class\_book\_ac1\_v2)

Figure 50: Introduction to classes - Instructions on how to create a class

You can add a new method to a class by adding a new function inside the class. For example, you can add the `initials` method to the `Person` class. The name of the function doesn't need to have any underscores in it. It only needs to start and end with double underscores if it is a special method like `__init__` or `__str__`. It does need to take the current object which is by convention referred to as `self`.

The following `Person` class has an `initials` method that returns a string with the first letter in the first name and the first letter in the last name in lowercase.

Save & Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 class Person:
2     """ Represents a person object """
3
4     # initializes the values in a new object called self
5     def __init__(self, first, last):
6         self.first = first # set first in self to the passed first
7         self.last = last  # set last in self to the passed last
8
9     # returns a string with information about the object self
10    def __str__(self):
11        return self.first + " " + self.last
12
13    # returns the first characters of the first and last name in lowercase
14    def initials(self):
15        return self.first[0].lower() + self.last[0].lower()
16
17 def main():
18     # calls the __init__ method
19     p1 = Person("Barbara", "Ericson")
20
21     # calls the __str__ method
22     print(p1)
23
24     # calls the initials method
25     print(p1.initials())
26
27 main()
28
```

Activity: 3 A class to represent a Person (class\_person\_init\_ac1\_v2)

Figure 51: Introduction to classes - worked example

## A.6 class-exp

Students are tasked with creating classes. One arrangement of the learning materials uses distractors while the other does not.

Create a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `len`. For example, `print(s)` when `s = Song('Respect',150)` would print "Respect, 150".

Drag from here

```
1 def __init__(self, title, len):
2     self.title = title
3     self.len = len
4
5 def __str__(self):
6     class Song:
7     return self.title + ", " + str(self.len)
```

Drop blocks here

Check Reset Help me

Parsons (Classes\_Basic\_Song\_nd\_pp)

Figure 52: class-exp - Define a song class with name and duration attributes

Create a class `Cat` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns "name, age". For example if `c = Cat("Fluffy", 3)` then `print(c)` should print "Fluffy, 3". Then define the `make_sound` method to return "Meow".

Drag from here

```
1 def make_sound(self):
2     self.name = name
3     self.age = age
4     return self.name + ", " + str(self.age)
5     return "Meow"
6
7 def __init__(self, name, age):
8     class Cat:
9     def __str__(self):
```

Drop blocks here

Check Reset Help me

Parsons (Classes\_Basic\_Cat\_nd\_pp)

Figure 53: class-exp - Define a cat class with name and age attributes and a method for making sound

Create a class `Book` with an `__init__` method that takes a `title` as a string and `page_len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `page_len`. For example, `print(b)` when `b = Book('Help', 300)` would print "Help, 300". Then add a `est_time` method that returns the estimated time to read the book. The estimated time to read a book is the number of pages multiplied by 1.5.

Drag from here

```

1 def est_time(self):
2 def __init__(self, title, page_len):
3 return self.title + ", " + str(self.page_len)
4 return 1.5 * self.page_len
5 self.title = title
  self.page_len = page_len
6 def __str__(self):
7 class Book:

```

Drop blocks here

Check Reset Help me

Parsons (Classes\_Basic\_Book\_nd\_pp)

Figure 54: *class-exp* - Define a book class with title and length attributes and a string-conversion method

Create a class `Account` with an `__init__` method that takes `id` and `balance` as numbers. Then create a `__str__` method that returns "id, balance". Next create a `deposit` method takes `amount` as a number and adds that to the `balance`. For example, if `a = Account(32, 100)` and `a.deposit(50)` is executed, `print(a)` should print "32, 150".

Drag from here

```

1 return str(self.id) + ", " + str(self.balance)
2 def deposit(self, amount):
3 def __str__(self):
4 def __init__(self, id, balance):
5 self.balance += amount
6 class Account:
7 self.id = id
  self.balance = balance

```

Drop blocks here

Check Reset Help me

Parsons (Classes\_Basic\_Account\_nd\_pp)

Figure 55: *class-exp* - Define a bank account class with identifier and balance attributes and a deposit method



Create a class `FortuneTeller` with an `__init__` method that takes a list of fortunes as strings and saves that as an attribute. Then create a `tell_fortune` method that returns one of the fortunes in the list at random.

Drag from here

```
1 import random
2 def __init__(self, fortunes):
3     self.fortunes = fortunes
4     return self.fortunes[index]
5 def tell_fortune(self):
6     last = len(self.fortunes) - 1
7     index = random.randint(0, last)
8     class FortuneTeller:
```

Drop blocks here

Check Reset Help me

Parsons (Classes\_Basic\_FortuneTeller\_nd\_pp)

Figure 56: *class-exp* - Define a fortune teller class which randomly selects a fortune from a predefined list

Please answer the following problems to the best of your ability without any outside help. You can stop working on a problem after you worked on it for about five minutes without solving it.

Create a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `len`. For example, `print(s)` when `s = Song('Respect',150)` would print "Respect, 150".

Drag from here

1a `def __str__():`

or

1b `def __str__(self):`

2a `my.title = title`  
`my.len = len`

or

2b `self.title = title`  
`self.len = len`

3 `class Song:`

4a `def __init__(title, len):`

or

4b `def __init__(self, title, len):`

5a `return self.title + ", " + str(self.len)`

or

5b `return title + ", " + str(len)`

Drop blocks here

Check

Reset

Help me

Parsons (Classes\_Basic\_Song\_wd\_pp\_v4)

Figure 57: *class-exp* - Define a song class with name and duration attributes (with distractors)

Create a class `Cat` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns "name, age". For example if `c = Cat("Fluffy", 3)` then `print(c)` should print "Fluffy, 3". Then define the `make_sound` method to return "Meow".

Drag from here

Drop blocks here

1a

return "Meow"

or

1b

return self."Meow"

2a

def make\_sound(self):

or

2b

def make\_sound():

3

class Cat:

4

def \_\_str\_\_(self):

5a

return self.name + ", " + self.age

or

5b

return name + ", " + age

6

self.name = name

self.age = age

7

def \_\_init\_\_(self, name, age):

Check

Reset

Help me

Parsons (Classes\_Basic\_Cat\_wd\_pp)

Figure 58: *class-exp* Define a cat class with name and age attributes and a method for making sound (with distractors)

Create a class `Book` with an `__init__` method that takes a `title` as a string and `page_len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `page_len`. For example, `print(b)` when `b = Book('Help', 300)` would print "Help, 300". Then add a `est_time` method that returns the estimated time to read the book. The estimated time to read a book is the number of pages multiplied by 1.5.

Drag from here

Drop blocks here

1 class Book:

2a return 1.5 \* self.page\_len

2b return self.1.5 \* self.page\_len

3 self.title = title  
self.page\_len = page\_len

4a def \_\_str\_\_(self):

4b def \_\_str\_\_():

5a def est\_time(self):

5b def est\_time():

6a return title + ", " + str(page\_len)

6b return self.title + ", " + str(self.page\_len)

7a def \_\_init\_\_(self, title, page\_len):

7b def \_\_init\_\_(title, page\_len):

or

or

or

or

or

or

or

or

Check Reset Help me

Parsons (Classes\_Basic\_Book\_wd\_pp\_v2)

Figure 59: *class-exp*- Define a book class with title and length attributes and with a string-conversion method (with distractors)

Create a class `Account` with an `__init__` method that takes `id` and `balance` as numbers. Then create a `__str__` method that returns "`id`, `balance`". Next create a `deposit` method takes `amount` as a number and adds that to the `balance`. For example, if `a = Account(32, 100)` and `a.deposit(50)` is executed, `print(a)` should print "`32, 150`".

Drag from here

Drop blocks here

1a `return str(self.id) + " , " + str(self.balance)`  
or  
1b `return self.id + " , " + str(balance)`  
2a `def __str__():`  
or  
2b `def __str__(self):`  
3a `self.balance += self.amount`  
or  
3b `self.balance += amount`  
4 `class Account:`  
5 `self.id = id`  
`self.balance = balance`  
6a `def __init__(id, balance):`  
or  
6b `def __init__(self, id, balance):`  
7 `def deposit(self, amount):`

Check Reset Help me

Parsons (Classes\_Basic\_Account\_wd\_pp\_v3)

Figure 60: *class-exp* - Define a bank account class with identifier and balance attributes and a deposit method (with distractors)

Create a class `FortuneTeller` with an `__init__` method that takes a list of fortunes as strings and saves that as an attribute. Then create a `tell_fortune` method that returns one of the fortunes in the list at random.

Drag from here

Drop blocks here

1 `class FortuneTeller:`  
2a `last = len(self.fortunes)`  
2b `last = len(self.fortunes) - 1`  
3a `def tell_fortune():`  
3b `def tell_fortune(self):`  
4 `import random`  
5 `index = random.randint(0, last)`  
6 `def __init__(self, fortunes):`  
7 `self.fortunes = fortunes`  
8a `return self.fortunes[index]`  
8b `return fortunes[index]`

Check Reset Help me

Parsons (Classes\_Basic\_FortuneTeller\_wd\_pp)

Figure 61: *class-exp* - Define a fortune teller class which randomly selects a fortune from a predefined list *class-exp* (with distractors)

Fix the class `Movie` that has an `__init__` method that takes a `title` as a string and `year` as a number and initializes these attributes in the current object. Also fix the `__str__` method to return the `title, year`. For example, `print(m)` when `m = Movie('Elvis',2022)` would print `"Elvis, 2022"`.

☐  
**Run**

Original - 1 of 1

**Show CodeLens****Share Code**

```
1 class Movie:
2
3     def __init__(title, year):
4         self.title = title
5         year = year
6
7     def str():
8         return title + ", " + year
9
10 m = Movie('Inception',2010)
11 print(m)
12
```

Activity: 1 ActiveCode (Classes\_Basic\_Movie\_fix\_v3\_ac)

Figure 62: *class-exp* - Posttest task for defining a movie class

Fix the class `Rectangle` with an `__init__` method that takes a `width` and `height` as numbers and initializes attributes with the same name in the current object. Then create a `__str__` method that returns `"width, height"` as a string. Next create an `total_area` method that takes a number of rectangles, `num` and returns `num` times `width` times `height`. For example, if `rec = Rectangle(15, 2)` and `rec.area(3)` is executed, it should print 90 ( $3 * 15 * 2$ ).

☐  
**Run**

Original - 1 of 1

**Show CodeLens****Share Code**

```
1
2 class Rectangle:
3     def __init__(width, height):
4         my.width = width
5         my.height = height
6
7     def __str__(self):
8         return width + ", " + height
9
10    def total_area(self, num):
11        return self.num * self.width * self.height
12
13 rec = Rectangle(15,2)
14 print(rec)
15 print(rec.total_area(3))
16
```

Activity: 1 ActiveCode (Classes\_Basic\_Rectangle\_ac\_fix\_v2)

Figure 63: *class-exp* - Posttest task for defining a rectangle class with a string-conversion method



Write a class `Horse` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns `"name, age"`. For example if `h = Horse("Midnight", 10)` then `print(c)` should print `Midnight, 10`. Then define the `make_sound` method to return `"Neigh"`.

☐

[Run](#) Original - 1 of 1 [Show CodeLens](#) [Share Code](#)

```
1
2 h = Horse("Midnight", 10)
3 print(h)
4 print(h.make_sound())
5
```

Activity: 1 ActiveCode (Classes\_Basic\_Horse\_v2\_ac)

Figure 64: *class-exp* - Posttest task for defining a horse class and a method for making sound

Write a class `GasTank` with an `__init__` method that takes a `max` and `curr_gas` as numbers. Then create a `__str__` method that returns `"max, curr_gas"`. Next create an `add_gas` method takes `amount` as a number and adds that to the `curr_gas`. For example, if `gt = GasTank(15, 2)` and `gt.add_gas(10)` is executed, `print(a)` should print "15, 12".

☐ Original - 1 of 1 Show CodeLens Share Code

```
1
2 gt = GasTank(15,2)
3 print(gt)
4 gt.add_gas(10)
5 print(gt)
6
```

Activity: 1 ActiveCode (Classes\_Basic\_GasTank\_ac)

Figure 65: *class-exp* - Posttest task for defining a gas tank class and a method to add gas

Fix the class `Dice` that has an `__init__` method that takes the number of sides, `num`, as a number and initializes that as an attribute. Also fix the `__str__` method to return `num` as a string. For example, `print(d)` when `d = Dice(6)` should print `6`. There is also a `roll` method that should return a random number from 1 to `num` (inclusive).

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 import random
2 class Dice:
3
4     def init(num):
5         self.num = num
6
7     def __str__():
8         return num
9
10    def roll()
11        random.randint(0,num-1)
12
13 d = Dice(6)
14 print(d)
15
```

Activity: 1 ActiveCode (Classes\_Basic\_Dice\_fix\_v2\_ac)

Figure 66: *class-exp* - Posttest task for defining a dice class with a given number of sides and a roll method

### A.7 *class-tog*

Similar to the previous activity, students are tasked with defining classes, with the same practice and posttest items. However, in this study, students are able to switch freely between a Parsons problem and code using a drop-down menu throughout the practice activities.

The screenshot displays the 'class-tog' activity interface. At the top, a 'Toggle Question:' section contains a dropdown menu with three options: 'Active Write Code - Classes\_Basic\_Song\_ac\_v2 (only this question will be graded)' (selected), 'Active Write Code - Classes\_Basic\_Song\_ac\_v2 (only this question will be graded)' (highlighted in blue), and 'Parsons Mixed-Up Code - Classes\_Basic\_Song\_pp\_v2'. Below the menu, a text box provides instructions: 'Write a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title, len`. For example, `print(s)` when `s = Song('Respect',150)` would print "Respect, 150".

Below the instructions, there is a progress bar and three buttons: 'Run' (green), 'Original - 1 of 1' (text), 'Show CodeLens' (dark grey), and 'Share Code' (teal). A code editor window shows the following code:

```
1
2 s = Song('Respect',150)
3 print(s)
4
5
```

At the bottom of the interface, it says 'Activity: 5 ActiveCode (Classes\_Basic\_Song\_ac\_v2)'.

Figure 67: Students have the ability to toggle between code and a Parsons problem

Write a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `len`. For example, `print(s)` when `s = Song('Respect',150)` would print `'Respect, 150'`.

Run Original - 1 of 1 Show CodeLens Share Code

```

1
2 s = Song('Respect',150)
3 print(s)
4
5

```

Activity: 5 ActiveCode (Classes\_Basic\_Song\_ac\_v2)

Create a class `Song` with an `__init__` method that takes a `title` as a string and `len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `len`. For example, `print(s)` when `s = Song('Respect',150)` would print `'Respect, 150'`.

Drag from here Drop blocks here

```

1 self.title = title
  self.len = len
2a def __str__(self):
or 2b def __str__():
3a def __init__(self, title, len):
or 3b def __init__(title, len):
4a return title + ", " + len
or 4b return self.title + ", " + str(self.len)
5a class Song:
or 5b class Song:

```

Check Reset Help me

Parsons (Classes\_Basic\_Song\_pp\_v2)

Figure 68: *class-tog* - Define a song class with name and duration attributes

Write a class `Cat` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns `'name: name, age: age'`. For example if `c = Cat('Fluffy', 3)` then `print(c)` should print `'name: Fluffy, age: 3'`. Then define the `make_sound` method to return `'Meow'`.

Run Original - 1 of 1 Show CodeLens Share Code

```

1
2 c = Cat('Fluffy', 3)
3 print(c)
4 print(c.make_sound())
5
6

```

Activity: 6 ActiveCode (Classes\_Basic\_Cat\_ac\_v2)

Create a class `Cat` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns `'name: name, age: age'`. For example if `c = Cat('Fluffy', 3)` then `print(c)` should print `'name: Fluffy, age: 3'`. Then define the `make_sound` method to return `'Meow'`.

Drag from here Drop blocks here

```

1 def __str__(self):
or 2a def make_sound(self):
or 2b def make_sound():
3 class Cat:
4a return f'name: {name}, age: {age}'
or 4b return f'name: {self.name}, age: {self.age}'
5 def __init__(self, name, age):
or 6a return self."Meow"
or 6b return "Meow"
7 self.name = name
  self.age = age

```

Check Reset Help me

Parsons (Classes\_Basic\_Cat\_pp\_v2)

Figure 69: *class-tog* - Define a cat class with name and age attributes and with a method for making sound

Write a class `Book` with an `__init__` method that takes a `title` as a string and `page_len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `page_len`. For example, `print(b)` when `b = Book('Black Beauty', 255)` would print 'Black Beauty, 255'. Then add a `est_time` method that takes a rate and returns the estimated time to read the book. The estimated time to read a book is the number of pages multiplied by the rate. For example, `b.est_time(1.5)` should return 382.5.

Run Original - 1 of 1 Show CodeLens Share Code

```

1
2 b = Book('Black Beauty', 255)
3 print(b)
4 print(b.est_time(1.5))
5
6

```

Activity 1 ActiveCode (Classes\_Basic\_Book\_ac)

Write a class `Book` with an `__init__` method that takes a `title` as a string and `page_len` as a number and initializes these attributes in the current object. Then define the `__str__` method to return the `title`, `page_len`. For example, `print(b)` when `b = Book('Black Beauty', 255)` would print 'Black Beauty, 255'. Then add a `est_time` method that takes a rate and returns the estimated time to read the book. The estimated time to read a book is the number of pages multiplied by the rate. For example, `b.est_time(1.5)` should return 382.5.

Drag from here Drop blocks here

```

1a def est_time(self, rate):
1b def est_time(rate):
2a return rate * self.page_len
2b return self.rate * self.page_len
3a def __str__(self):
3b def __str__():
4a def __init__(self, title, page_len):
4b def __init__(title, page_len):
5a return self.title + ", " + str(self.page_len)
5b return title + ", " + str(page_len)
6 class Book:
7 self.title = title
  self.page_len = page_len

```

Check Reset Help me

Parsons (Classes\_Basic\_Book\_wd\_pp\_v3)

Figure 70: *class-tog* - Define a book class with title and length attributes and a string-conversion method

Create a class `Account` with an `__init__` method that takes `id` and `balance` as numbers. Then create a `__str__` method that returns "id, balance". Next create a `deposit` method takes `amount` as a number and adds that to the `balance`. For example, if `a = Account(32, 100)` and `a.deposit(50)` is executed, `print(a)` should print "32, 150".

Run Original - 1 of 1 Show CodeLens Share Code

```

1
2 a = Account(32, 100)
3 a.deposit(50)
4 print(a)
5
6

```

Activity 7 ActiveCode (Classes\_Basic\_Account\_ac\_v2)

Create a class `Account` with an `__init__` method that takes `id` and `balance` as numbers. Then create a `__str__` method that returns "id, balance". Next create a `deposit` method takes `amount` as a number and adds that to the `balance`. For example, if `a = Account(32, 100)` and `a.deposit(50)` is executed, `print(a)` should print "32, 150".

Drag from here Drop blocks here

```

1a def __init__(self, id, balance):
1b def __init__(self, id, balance)
2 class Account:
3a return f"{self.id}, {self.balance}"
3b return f"{id}, {balance}"
4a self.balance += self.amount
4b self.balance += amount
5 def deposit(self, amount):
6 def __str__(self):
7 self.id = id
  self.balance = balance

```

Check Reset Help me

Parsons (Classes\_Basic\_Account\_pp\_v2)

Figure 71: *class-tog* - Define a bank account class with identifier and balance attributes and a deposit method

Write a class `FortuneTeller` with an `__init__` method that takes a list of fortunes as strings and saves that as an attribute. Then create a `tell_fortune` method that returns one of the fortunes in the list at random.

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 f = FortuneTeller(["You will get an A", "You will have a great day", "You will fall", "You will be rich"])
2
3 for i in range(5):
4     print(f.tell_fortune())
5
6
```

Activity: 8 ActiveCode (Classes\_Basic\_FortuneTeller\_ac\_v2)

Create a class `FortuneTeller` with an `__init__` method that takes a list of fortunes as strings and saves that as an attribute. Then create a `tell_fortune` method that returns one of the fortunes in the list at random.

Drag from here

Drop blocks here

1 index = random.randint(0, last)

2a return self.fortunes[index]

2b return fortunes[index]

3 import random

4 self.fortunes = fortunes

5 class FortuneTeller:

6a last = len(self.fortunes) - 1

6b last = len(self.fortunes)

7a def tell\_fortune():

7b def tell\_fortune(self):

8 def \_\_init\_\_(self, fortunes):

Check

Reset

Help me

Parsons (Classes\_Basic\_FortuneTeller\_pp\_v2)

Figure 72: *class-tog* - Define a fortune teller class which randomly selects a fortune from a predefined list (with distractors)

Fix the class `Movie` that has an `__init__` method that takes a `title` as a string and `year` as a number and initializes these attributes in the current object. Also fix the `__str__` method to return the `title, year`. For example, `print(m)` when `m = Movie('Elvis',2022)` would print `"Elvis, 2022"`.

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 class Movie:
2
3     def __init__(title, year):
4         self.title = title
5         year = year
6
7     def str():
8         return title + ", " + year
9
10 m = Movie('Inception',2010)
11 print(m)
12
13
```

Activity: 1 ActiveCode (Classes\_Basic\_Movie\_fix\_v3\_ac)

Figure 73: *class-tog* - Posttest task for defining a movie class



Fix the class `Rectangle` with an `__init__` method that takes a `width` and `height` as numbers and initializes attributes with the same name in the current object. Then create a `__str__` method that returns `"width, height"` as a string. Next create an `total_area` method that takes a number of rectangles, `num` and returns `num` times `width` times `height`. For example, if `rec = Rectangle(15, 2)` and `rec.area(3)` is executed, it should print 90 ( $3 * 15 * 2$ ).

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1
2 class Rectangle:
3     def __init__(width, height):
4         my.width = width
5         my.height = height
6
7     def __str__(self):
8         return width + ", " + height
9
10    def total_area(self, num):
11        return self.num * self.width * self.height
12
13 rec = Rectangle(15,2)
14 print(rec)
15 print(rec.total_area(3))
16
```

Activity: 1 ActiveCode (Classes\_Basic\_Rectangle\_ac\_fix\_v2)

Figure 74: *class-tog* - Post-posttest task for defining a rectangle class and string-conversion method

Write a class `Horse` with an `__init__` method that takes `name` as a string and `age` as a number and initializes these attributes in the current object. Next create the `__str__` method that returns `"name, age"`. For example if `h = Horse("Midnight", 10)` then `print(c)` should print `Midnight, 10`. Then define the `make_sound` method to return `"Neigh"`.

☐ Run Original - 1 of 1 Show CodeLens Share Code

```
1
2 h = Horse("Midnight", 10)
3 print(h)
4 print(h.make_sound())
5
6
```

Activity: 1 ActiveCode (Classes\_Basic\_Horse\_v2\_ac)

Figure 75: *class-tog* - Post-posttest task for defining a horse class and a method for making sound

Write a class `GasTank` with an `__init__` method that takes a `max` and `curr_gas` as numbers. Then create a `__str__` method that returns `"max, curr_gas"`. Next create an `add_gas` method takes `amount` as a number and adds that to the `curr_gas`. For example, if `gt = GasTank(15, 2)` and `gt.add_gas(10)` is executed, `print(a)` should print "15, 12".

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1
2 gt = GasTank(15,2)
3 print(gt)
4 gt.add_gas(10)
5 print(gt)
6
7
```

Activity: 1 ActiveCode (Classes\_Basic\_GasTank\_ac)

Figure 76: *class-tog* - Posttest task for defining a gas tank class and a method for adding gas

Fix the class `Dice` that has an `__init__` method that takes the number of sides, `num`, as a number and initializes that as an attribute. Also fix the `__str__` method to return `num` as a string. For example, `print(d)` when `d = Dice(6)` should print `6`. There is also a `roll` method that should return a random number from 1 to `num` (inclusive).

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 import random
2 class Dice:
3
4     def init(num):
5         self.num = num
6
7     def __str__():
8         return num
9
10    def roll()
11        random.randint(0,num-1)
12
13 d = Dice(6)
14 print(d)
15
```

Activity: 1 ActiveCode (Classes\_Basic\_Dice\_fix\_v2\_ac)

Figure 77: *class-tog* - Posttest task for defining a dice class with a given number of sides and a roll method

## A.8 p3dnd

The *p3dnd* study is similar to *p3pt* but is intended to have harder practice and posttest problems. The two conditions are Parsons problems with and without distractors.

Create the function `front_back(str, start, end)` that takes three strings and returns a string based on the following conditions.

- If `str` contains `start` at the beginning of the string return `"s"`.
- if `str` contains `end` at the end of the string return `"e"`.
- If `str` contains `start` at the beginning and `end` at the end then return `"s_e"`.
- Otherwise return `"n"`.

Example Input	Expected Output
<code>front_back("Opening time", "Open", "noon")</code>	<code>"s"</code>
<code>front_back("Afternoon", "Open", "noon")</code>	<code>"e"</code>
<code>front_back("Open at noon", "Open", "noon")</code>	<code>"s_e"</code>
<code>front_back("Closed", "Open", "noon")</code>	<code>"n"</code>
<code>front_back("It is noon now", "open", "noon")</code>	<code>"n"</code>

Drag from here

1 `def front_back(str, start, end):`

2 `if str.startswith(start) and str[last:] == end:`

3 `return "s_e"`

4 `return "s"`

5 `return "e"`

6a `last = len(end)`

or

6b `last = len(end) * -1`

7 `return "n"`

8a `elif str[last] == end:`

or

8b `elif str[last:] == end:`

9a `elif str.startswith(start):`

or

9b `elif str.starts(start):`

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-front-back-wd)

Figure 78: *p3dnd* - return a character based on what is at the beginning and end of a string (with distractors)

96

Create a function, `bobThere(str)` that takes a string, `str`. It returns `True` if `str` contains a "bob" string, but where the middle 'o' char can be any char. Otherwise it returns `False`.

Example Input	Expected Output
<code>bobThere("abcbob")</code>	<code>True</code>
<code>bobThere("b9b")</code>	<code>True</code>
<code>bobThere("bac")</code>	<code>False</code>

*Drag from here*

{

1a

for i in range(len(str)):

}

1b

for i in range(len(str)-2):

2

def bobThere(str):

3

return False

{

4a

if str[i] == 'b' and str[i+2] == 'b':

}

4b

if str[i] == 'b' and str[i] == 'b':

5

return True

*Drop blocks here*

Check
Reset
Help me

Parsons (p3dndta-bob-there-wd)

Figure 79: *p3dnd* - return true if a string contains "b\*b" where '\*' can be any character (with distractors)

Create a function `sum13(nums)` that takes a list of numbers, `nums` and returns the sum of the numbers in the list. However, the number 13 is very unlucky, so do not add it or the number that comes immediately after a 13 to the sum. Return `0` if `nums` is the empty list.

#### Example Input

#### Expected Output

<code>sum13([13,1,2])</code>	<code>2</code>
<code>sum13([1,13])</code>	<code>1</code>
<code>sum13([4, 13, 8])</code>	<code>4</code>
<code>sum13([13, 1, 13, 3, 2])</code>	<code>2</code>

Drag from here

1

def sum\_13(nums):

2

found\_13 = False

3

if found\_13:

4

total += num

5

found\_13 = True

6

else:

7a

total = 0

7b

found\_13 = True

8a

total = 0

8b

found\_13 = False

8a

return Total

8b

return total

9

for num in nums:

10a

elif num == 13:

10b

elif num == 13:

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-sum13-wd)

Figure 80: *p3dnd* - return the sum of a list of numbers, but ignore 13 and any number after a 13 (with distractors)

Create a function `twoSum(nums, target)` that takes a list of integers `nums` and an integer `target` and returns the indices of the two numbers such that they add up to `target`. If no two numbers add up to `target`, it returns an empty list. Assume that each input has exactly one solution, and you may not use the same element twice.

Example Input	Expected Output
<code>twoSum([2,7,11,15], 9)</code>	<code>[0, 1]</code>
<code>twoSum([2,7,11,15], 13)</code>	<code>[0, 2]</code>
<code>twoSum([2,7,11,15], 5)</code>	<code>[]</code>

Drag from here

1a

for j in range(i, len(nums)):

or

1b

for j in range(i+1, len(nums)):

or

2a

if nums[i] + nums[i] == target:

2b

if nums[i] + nums[j] == target:

3

return [i, j]

4

def twoSum(nums, target):

5

return []

6

for i in range(len(nums)):

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-two-sum-wd)

Figure 81: *p3dnd* - returns the indices of two numbers in a list that add up to a passed target value (with distractors)



Create the function `twoTwo(nums)` that takes a list of numbers, `nums` and returns `True` if every 2 that appears in the list is next to another 2. Otherwise it returns `False`.

Example Input	Expected Output
<code>twoTwo([4, 2, 2, 3])</code>	<code>True</code>
<code>twoTwo([2, 2, 4])</code>	<code>True</code>
<code>twoTwo([2, 2, 4, 2])</code>	<code>False</code>

Drag from here

1a

elif nums[i+1] == 2:

or

1b

elif i < len(nums) - 1 and nums[i+1] == 2:

2

for i in range(len(nums)):

3

if nums[i] == 2:

4a

break

or

4b

continue

5

return True

6a

break

or

6b

continue

7

def twoTwo(nums):

8a

if nums[i-1] == 2:

or

8b

if i > 0 and nums[i-1] == 2:

9

else:

10

return False

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-two-two-wd)

Figure 82: *p3dnd* - return true if every two in a list of numbers is next to another two (with distractors)

Create a function `isPalindrome(x)` that takes an integer, `x`, and returns `True` if `x` is a palindrome, and `False` otherwise. An integer is a palindrome if the digits read the same backwards as forwards.

Example Input	Expected Output
<code>isPalindrome(121)</code>	<code>True</code>
<code>isPalindrome(888)</code>	<code>True</code>
<code>isPalindrome(678)</code>	<code>[]</code>

Drag from here

1 | while left < right:

2a | strx = str(x)  
left = 0  
right = len(strx) - 1

2b | left = 0  
right = len(strx)

3a | left += 1  
right -= 1

3b | left -= 1  
right += 1

4 | return True

5 | if strx[left] != strx[right]:

6 | def isPalindrome(x):

7 | return False

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-palindrome-number-wd)

Figure 83: p3dnd - returns true if the digits in a number are a palindrome (with distractors)

101

Create the function `front_back(str, start, end)` that takes three strings and returns a string based on the following conditions.

- If `str` contains `start` at the beginning of the string return `"s"`.
- if `str` contains `end` at the end of the string return `"e"`.
- If `str` contains `start` at the beginning and `end` at the end then return `"s_e"`.
- Otherwise return `"n"`.

Example Input	Expected Output
<code>front_back("Opening time", "Open", "noon")</code>	<code>"s"</code>
<code>front_back("Afternoon", "Open", "noon")</code>	<code>"e"</code>
<code>front_back("Open at noon", "Open", "noon")</code>	<code>"s_e"</code>
<code>front_back("Closed", "Open", "noon")</code>	<code>"n"</code>
<code>front_back("It is noon now", "open", "noon")</code>	<code>"n"</code>

Drag from here

1

`return "n"`

2

`return "e"`

3

`if str.startswith(start) and str[last:] == end:`

4

`elif str[last:] == end:`

5

`last = len(end) * -1`

6

`return "s_e"`

7

`def front_back(str, start, end):`

8

`elif str.startswith(start):`

9

`return "s"`

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-front-back-nd)

Figure 84: *p3dnd* - return a character based on what is at the beginning and end of a string

Create a function, `bobThere(str)` that takes a string, `str`. It returns `True` if `str` contains a "bob" string, but where the middle 'o' char can be any char. Otherwise it returns `False`.

Example Input	Expected Output
<code>bobThere("abcbob")</code>	<code>True</code>
<code>bobThere("b9b")</code>	<code>True</code>
<code>bobThere("bac")</code>	<code>False</code>

Drag from here

```
1 | for i in range(len(str)-2):  
2 | def bobThere(str):  
3 | if str[i] == 'b' and str[i+2] == 'b':  
4 | return True  
5 | return False
```

Drop blocks here

Check Reset Help me

Parsons (p3dndta-bob-there-nd)

Figure 85: *p3dnd* - return true if a string contains "b\*b" where "\*" can be any character

Create a function `sum13(nums)` that takes a list of numbers, `nums` and returns the sum of the numbers in the list. However, the number 13 is very unlucky, so do not add it or the number that comes immediately after a 13 to the sum. Return `0` if `nums` is the empty list.

Example Input	Expected Output
<code>sum13([13,1,2])</code>	<code>2</code>
<code>sum13([1,13])</code>	<code>1</code>
<code>sum13([4, 13, 8])</code>	<code>4</code>
<code>sum13([13, 1, 13, 3, 2])</code>	<code>2</code>

Drag from here

```
1 def sum_13(nums):
2     elif num == 13:
3     total = 0
4     found_13 = False
5     total += num
6     if found_13:
7     found_13 = True
8     return total
9     for num in nums:
10    found_13 = False
11    else:
```

Drop blocks here

Check Reset Help me

Parsons (p3dndta-sum13-nd)

Figure 86: *p3dnd* - return a sum of a list of numbers, but ignore 13 and any number after a 13

Create a function `twoSum(nums, target)` that takes a list of integers `nums` and an integer `target` and returns the indices of the two numbers such that they add up to `target`. If no two numbers add up to `target`, it returns an empty list. Assume that each input has exactly one solution, and you may not use the same element twice.

Example Input	Expected Output
<code>twoSum([2,7,11,15], 9)</code>	<code>[0, 1]</code>
<code>twoSum([2,7,11,15], 13)</code>	<code>[0, 2]</code>
<code>twoSum([2,7,11,15], 5)</code>	<code>[]</code>

Drag from here

```
1 def twoSum(nums, target):  
2     for i in range(len(nums)):  
3         return []  
4     if nums[i] + nums[j] == target:  
5         return [i, j]  
6     for j in range(i+1, len(nums)):
```

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-two-sum-nd)

Figure 87: *p3dnd* - returns the indices of two numbers in a list that add up to a passed target value

Create the function `twoTwo(nums)` that takes a list of numbers, `nums` and returns true if every 2 that appears in the list is next to another 2. Otherwise it returns `False`.

Example Input	Expected Output
<code>twoTwo([4, 2, 2, 3])</code>	<code>True</code>
<code>twoTwo([2, 2, 4])</code>	<code>True</code>
<code>twoTwo([2, 2, 4, 2])</code>	<code>False</code>

Drag from here

1 | `else:`

2 | `if nums[i] == 2:`

3 | `elif i < len(nums) - 1 and nums[i+1] == 2:`

4 | `for i in range(len(nums)):`

5 | `continue`

6 | `return True`

7 | `continue`

8 | `return False`

9 | `def twoTwo(nums):`

10 | `if i > 0 and nums[i-1] == 2:`

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-two-two-nd)

Figure 88: *p3dnd* - return true if every two in a list of numbers is next to another two

106

Create a function `isPalindrome(x)` that takes an integer, `x`, and returns `True` if `x` is a palindrome, and `False` otherwise. An integer is a palindrome if the digits read the same backwards as forwards.

Example Input	Expected Output
<code>isPalindrome(121)</code>	<code>True</code>
<code>isPalindrome(888)</code>	<code>True</code>
<code>isPalindrome(678)</code>	<code>[]</code>

Drag from here

1

`return False`

2

`def isPalindrome(x):`

3

`strx = str(x)`  
 `left = 0`  
 `right = len(strx) - 1`

4

`return True`

5

`if strx[left] != strx[right]:`

6

`left += 1`  
 `right -= 1`

7

`while left < right:`

Drop blocks here

Check

Reset

Help me

Parsons (p3dndta-palindrome-number-nd)

Figure 89: *p3dnd* - returns true if the digits in a number are a palindrome



Write the function `upper_center(str)` to return the passed string `str` with the middle character(s) in uppercase.

- If `str` has an odd length, uppercase the middle character.
- If `str` has an even length, uppercase the middle two characters.
- If `str` has less than 3 characters then return `str`.

#### Example Input

#### Expected Output

`upper_center('abc')`

`'aBc'`

`upper_center('abcd')`

`'aBCd'`

`upper_center('a')`

`'a'`

Run

Original - 1 of 1

Share Code

```
1 def upper_center(str):  
2  
3
```

Activity: 29 ActiveCode (p3dnd\_upper\_center)

Figure 90: *p3dnd* - return a string with the center characters in uppercase or just the string if the length is less than 3

Write a function `is_descending(nums)` to do the following.

- Return `True` if the numbers in the list `nums` are sorted in descending order.
- Otherwise return `False`.
- If the list `nums` has less than two numbers in it return `True`.

#### Example Input

#### Expected Output

`is_descending([2,3,4])`

`False`

`is_descending([1])`

`True`

`is_descending([4,3,2])`

`True`

Run

Original - 1 of 1

Share Code

```
1 def is_descending(nums):
2     #write your code here
3
4
5
6 print(is_descending([2, 3, 4]))
7 print(is_descending([3]))
8 print(is_descending([4, 3, 2]))
9
10
```

Activity: 30 ActiveCode (p3dnd\_is\_descending\_ac)

Figure 91: *p3dnd* - return true if the numbers in a list are sorted in descending order

Fix the `sum67` function below that takes a list of numbers and returns the total of the numbers in the list except for all the numbers whose position is between a 6 and 7 in the list (inclusive).

**Example Input**

**Expected Output**

`sum67([1,2])`

3

`sum67([2, 6, 8, 7, 2])`

4

`sum67([3, 6, 7])`

3

Run

Original - 1 of 1

Show CodeLens

Share Code

```
1 def sum67(nums):
2     total = 0                # initialize the total
3     found_6 = True           # initialize a Boolean flag
4     for num in nums:         # loop through the items in a list
5         if found_6 && num == 7:
6             found_6 = False # set the Boolean flag to false
7         elif num == 6:
8             found_6 = True  # set the Boolean flag to True
9         elif found_6:
10            continue        # go back to the top of the loop
11        else:
12            total += num     # add num to total
13        return total        # return the total
14
15
```

Activity: 31 ActiveCode (p3dnd\_sum67\_fix)

Figure 92: *p3dnd* - returns the total of all numbers in a list except those inclusively between a 6 and 7