# Programming Revision Worksheet

## 1. What are the contents of the list lst after the following code is executed?

**```python  
lst = [2\*x + y for x in range(3) for y in range(3, 6)]  
```**  
\*\*Options:\*\*

- [3, 4, 5, 6, 7, 8, 9]  
- [3, 4, 5, 5, 6, 7, 7, 8, 9]  
- [6, 7, 8, 8, 9, 10, 10, 11, 12]  
- [3, 5, 7, 4, 6, 8, 5, 7, 9]  
  
**\*\*Explanation:\*\***  
This code creates a list using numbers from two groups:  
- \*\*range(3):\*\* Picks numbers for `x`: `0, 1, 2`.  
- \*\*range(3, 6):\*\* Picks numbers for `y`: `3, 4, 5`.  
- \*\*Formula `2\*x + y:`\*\* Does math with `x` and `y`.  
  
\*\*Step-by-step Calculation:\*\*  
- When `x = 0`: `2\*0 + 3 = 3`, `2\*0 + 4 = 4`, `2\*0 + 5 = 5`  
- When `x = 1`: `2\*1 + 3 = 5`, `2\*1 + 4 = 6`, `2\*1 + 5 = 7`  
- When `x = 2`: `2\*2 + 3 = 7`, `2\*2 + 4 = 8`, `2\*2 + 5 = 9`

**Answer:**

\*\*Final list:\*\* `[3, 4, 5, 5, 6, 7, 7, 8, 9]`.

## 2. Which of the following are TRUE about object-oriented programming (OOP)?

\*\*Options:\*\*

- The `self` parameter refers to the object instance.  
- Encapsulation protects data.  
- Overridden methods match name and parameters.  
- The `super()` function lets us call methods from the parent class.  
  
**\*\*Explanation:\*\***  
- \*\*`self:`\*\* Imagine a robot saying 'I'll talk about myself!' In Python, `self` refers to the object itself.  
- \*\*`super():`\*\* Lets the child (like a young robot) use something from its parent (the main robot model).

**Answer:**

- The `self` parameter refers to the object instance.  
- The `super()` function lets us call methods from the parent class.

## 4. Adding to a List

\*\*Which methods can add elements to a list?\*\*  
  
\*\*Options:\*\*

- `insert()`  
- `extend()`  
- `append()`  
- `add()`  
  
**\*\*Explanation:\*\***  
- \*\*`append():`\*\* Adds one thing to the end of a list. Example:  
```python  
lst = [1, 2]  
lst.append(3) # [1, 2, 3]  
```  
- \*\*`insert():`\*\* Adds one thing at a specific position. Example:  
```python  
lst = [1, 2]  
lst.insert(1, 99) # [1, 99, 2]  
```  
- \*\*`extend():`\*\* Adds many things from another list. Example:  
```python  
lst = [1, 2]  
lst.extend([3, 4]) # [1, 2, 3, 4]  
```  
- \*\*`add():`\*\* Is not for lists-it's for sets.

**Answer:**

**All options but the add()**

## 5. Nested Loops

\*\*What will this code produce?\*\*  
**```python  
result = [(x, y) for x in range(3) for y in 'hi']  
print(result)  
```**  
  
\*\*Options:\*\*

- [(0, 'hi'), (1, 'hi'), (2, 'hi')]  
- [(0, 'h'), (0, 'i'), (1, 'h'), (1, 'i'), (2, 'h'), (2, 'i')]  
- [(0, 'h'), (1, 'h'), (2, 'h'), (0, 'i'), (1, 'i'), (2, 'i')]  
- [(0, 'h'), (1, 'i'), (2, 'h'), (0, 'i'), (1, 'h'), (2, 'i')]  
  
**\*\*Explanation:\*\***  
This code creates pairs by combining numbers with letters:  
- \*\*range(3):\*\* Picks numbers: `0, 1, 2`.  
- \*\*'hi':\*\* Picks letters: `'h', 'i'`.  
  
\*\*Step-by-step:\*\*  
- When `x = 0`: Combine with `'h'` and `'i'` -> `(0, 'h')`, `(0, 'i')`.  
- When `x = 1`: Combine with `'h'` and `'i'` -> `(1, 'h')`, `(1, 'i')`.  
- When `x = 2`: Combine with `'h'` and `'i'` -> `(2, 'h')`, `(2, 'i')`.

**Answer:**

\*\*Final list:\*\* `[(0, 'h'), (0, 'i'), (1, 'h'), (1, 'i'), (2, 'h'), (2, 'i')]`.

**6. Filtering Numbers**

\*\*What is the output of this code?\*\*  
**```python  
lst = []  
for i in range(10):  
 if (i % 2 == 0) and (i % 3 != 0):  
 lst.append(i)  
print(lst)  
```**  
  
\*\*Options:\*\*

- [2, 6, 8]  
- [2, 4, 8]  
- [4, 6, 8]  
- [2, 3, 6]  
  
**\*\*Explanation:\*\***  
The code adds numbers to a list if they meet two rules:  
- \*\*`i % 2 == 0:`\*\* The number is even (e.g., 2, 4, 6).  
- \*\*`i % 3 != 0:`\*\* The number is not divisible by 3 (e.g., not 3, 6, 9).  
  
\*\*Step-by-step:\*\*  
- Check numbers from 0 to 9:  
 - `0`: Even but divisible by 3 -> No.  
 - `2`: Even and not divisible by 3 -> Yes.  
 - `4`: Even and not divisible by 3 -> Yes.  
 - `6`: Even but divisible by 3 -> No.  
 - `8`: Even and not divisible by 3 -> Yes.

**Answer:**

\*\*Final list:\*\* `[2, 4, 8]`.

**7. Encapsulation**

\*\*What is the main purpose of encapsulation?\*\*  
  
\*\*Options:\*\*

- Makes a list faster  
- Hides data and functions  
- Creates a new object  
- Changes a variable’s value  
  
**\*\*Explanation:\*\***  
Encapsulation is like putting toys in a box and locking it. Only those with the right key (method) can open the box and use the toys (data).  
It keeps information \*\*safe\*\* and ensures only certain things can access it.

**Answer:**

Hides data and functions

**8. String Formatting**

\*\*What does this code print?\*\*  
**```python  
x = 2  
a = 'This '  
b = 'is a test '  
c = 'Resit {}'.format(x)  
d = a+b+c  
print(d)  
```**  
  
\*\*Options:\*\*

- This is a test Resit x  
- This is a test Resit 3  
- This is a test Resit 2  
- This is a test Resit  
  
**\*\*Explanation:\*\***  
- \*\*`{}`\*\* in `'Resit {}'` is a placeholder.  
- \*\*`.format(x)`\*\* replaces the placeholder with `x` (which is `2`).  
- Add all strings together: `'This ' + 'is a test ' + 'Resit 2'`.

**Answer:**

\*\*Final result:\*\* `'This is a test Resit 2'`.

**10. Which of the following is/are the correct way/s to create a Python dictionary?**

\*\*Options:\*\*

1- [ ] `dict1 = []`

2- ```python  
 days = ['Mon', 'Tue', 'Wed']  
 mode = ['face-to-face', 'online', 'one-on-one']  
 dictionary = dict(zip(days, mode))  
 ```

3- `dict2 = dict()`

4- `dict3 = dict2.copy()`

**\*\*Explanation:\*\***  
- \*\*`dict(zip(days, mode)):`\*\* Combines two lists into a dictionary where one list becomes the keys, and the other becomes the values.  
- \*\*`dict():`\*\* Creates an empty dictionary.  
- \*\*`dict2.copy():`\*\* Copies the contents of one dictionary into another.  
- \*\*`dict1 = []:`\*\* This creates a list, not a dictionary.

**Answer:**

ANY BUT ANSWER 1: - \*\*`dict1 = []:`\*\* This creates a list, not a dictionary.