**1. Create a list called years\_list, starting with the year of your birth, and each year thereafter until**

**the year of your fifth birthday. For example, if you were born in 1980. the list would be years\_list =**

**[1980, 1981, 1982, 1983, 1984, 1985].**

Absolutely! Here's how to create that list in Python:

Python

def create\_years\_list(birth\_year):

"""Creates a list of years from birth year to fifth birthday."""

years\_list = []

for year in range(birth\_year, birth\_year + 6):

years\_list.append(year)

return years\_list

# Example usage: Assuming you were born in 1980

birth\_year = 1980

my\_years\_list = create\_years\_list(birth\_year)

print(my\_years\_list) # Output: [1980, 1981, 1982, 1983, 1984, 1985]

**Explanation:**

1. **The create\_years\_list function:**
   * It takes your birth\_year as input.
   * It initializes an empty list called years\_list.
   * It uses a for loop to iterate from your birth year up to (and including) the year of your fifth birthday.
   * For each year, it uses years\_list.append(year) to add that year to the list.
   * Finally, it returns the completed list.
2. **Example Usage:**
   * We set birth\_year to 1980.
   * We call the create\_years\_list function to get your list.
   * We print the resulting list.

**2. In which year in years\_list was your third birthday? Remember, you were 0 years of age for your first year.**

Absolutely! Here's the Python code to do that, along with explanations:

**Code:**

Python

def create\_years\_list(birth\_year):

"""Creates a list of years from birth year to fifth birthday"""

years\_list = [year for year in range(birth\_year, birth\_year + 6)]

return years\_list

# Example usage (Assume your birth year is 1980)

birth\_year = 1980

years\_list = create\_years\_list(birth\_year)

print(years\_list) # Output: [1980, 1981, 1982, 1983, 1984, 1985]

# Find third birthday year

third\_birthday\_year = years\_list[2] # Remember, lists start indexing at 0

print("Your third birthday was in:", third\_birthday\_year) # Output: 1982

**Explanation:**

1. **create\_years\_list(birth\_year):**
   * This function takes your birth year as input.
   * It uses a list comprehension to generate the list of years from your birth year up to the year of your fifth birthday.
2. **Example Usage:**
   * We set birth\_year to 1980.
   * We call the create\_years\_list function to generate the list.
   * The years\_list would be [1980, 1981, 1982, 1983, 1984, 1985].
3. **Finding the Third Birthday:**
   * Python lists are zero-indexed (the first item is at position 0).
   * Since your third birthday is the third year in the list, we access it using years\_list[2].

**3.In the years list, which year were you the oldest?**

Absolutely! Here's how you would do this in Python, along with explanations:

**Creating the list**

Python

year\_of\_birth = 2000 # Replace with your actual year of birth

years\_list = list(range(year\_of\_birth, year\_of\_birth + 6))

print(years\_list)

**Explanation:**

1. **year\_of\_birth = 2000:** You store your birth year in a variable.
2. **list(range(year\_of\_birth, year\_of\_birth + 6)):**
   * The range function generates a sequence of numbers starting from your birth year up to (but not including) your birth year + 6. This covers your first six years.
   * The list function converts this sequence into a list.
3. **print(years\_list):** This displays your years\_list.

**Finding your third birthday and oldest year**

Python

# In which year in years\_list was your third birthday?

third\_birthday\_year = years\_list[2]

print(f"Year of third birthday: {third\_birthday\_year}")

# In the years list, which year were you the oldest?

oldest\_year = years\_list[-1]

print(f"Year you were the oldest: {oldest\_year}")

**Explanation:**

* **Lists are zero-indexed:** In Python, list elements start at index 0. So the first year is at index 0, the second at index 1, and your third birthday year is at index 2.
* **years\_list[-1]:** The index -1 gives you the last element in the list, which is the year you were oldest within that timeframe.

**4. Make a list called things with these three strings as elements: “mozzarella” “Cinderella”,“salmonella”**

Absolutely! Here's the Python code to create that list:

Python

things = ["mozzarella", "Cinderella", "salmonella"]

print(things)

**Explanation:**

* **things = [...]**: This creates a list named things. The square brackets [] are used to define lists in Python.
* **"mozzarella", "Cinderella", "salmonella"**: These are the string elements you want to include in your list, separated by commas. Strings are enclosed in quotation marks.

**6. Make a surprise list with the elements “Groucho”, “Chico” and “Harpo”**

Sure, you can create a surprise list with those elements in Python like this:

```python

surprise\_list = ["Groucho", "Chico", "Harpo"]

print("Surprise list:", surprise\_list)

```

This code will output:

```

Surprise list: ['Groucho', 'Chico', 'Harpo']

```

**7. Lowercase the last element of the surprise list, reverse it, and then capitalize it.**

You can achieve this in Python with the following code:

surprise\_list = ["Groucho", "Chico", "Harpo"]

last\_element = surprise\_list[-1].lower()[::-1].capitalize()

surprise\_list[-1] = last\_element

print(surprise\_list)

This code first creates the surprise list with the elements "Groucho", "Chico", and "Harpo". Then, it takes the last element, converts it to lowercase, reverses it, and capitalizes it. Finally, it updates the last element of the surprise list with this modified value and prints the resulting list.

**8. Make an English-to-French dictionary called e2f and print it. Here are your starter words: dog is chien, cat is chat, and walrus is morse.**

Here's the Python code to create the English-to-French dictionary and print it:

Python

# Create the English-to-French dictionary

e2f = {

"dog": "chien",

"cat": "chat",

"walrus": "morse"

}

# Print the dictionary in a readable format

for english\_word, french\_word in e2f.items():

print(f"{english\_word} is {french\_word}")

**Explanation:**

1. **Creating the dictionary:**
   * We use curly braces {} to create a dictionary named e2f.
   * Each word pair is a key-value item within the dictionary:
     + The English word is the key (e.g., "dog")
     + The French translation is the value (e.g., "chien")
2. **Printing the dictionary:**
   * We use a for loop to iterate through each key-value pair in the e2f.items()
   * f-strings are used to format the output nicely, making it more readable.

**Output:**

dog is chien

cat is chat

walrus is morse

**9. Write the French word for walrus in your three-word dictionary e2f.**

Here's how you can do this in Python:

**1. Create the dictionary:**

Python

e2f = {

"dog": "chien",

"cat": "chat",

"walrus": "morse"

}

**2. Print the dictionary:**

Python

print(e2f)

**3. Get the French word for "walrus":**

Python

french\_walrus = e2f["walrus"]

print(french\_walrus) # This will output "morse"

**Explanation:**

**Lines 1-4:**

* A dictionary named e2f is created.
* The dictionary uses curly braces {}.
* Inside the braces, you have key-value pairs separated by colons :.
  + The keys are the English words ("dog", "cat", "walrus").
  + The values are the corresponding French translations ("chien", "chat", "morse").

**Line 6:**

* print(e2f) will display the entire dictionary content in the following format:
* {'dog': 'chien', 'cat': 'chat', 'walrus': 'morse'}

**Lines 8-9:**

* e2f["walrus"] accesses the value associated with the key "walrus". In this case, the value is "morse".
* print(french\_walrus) outputs the word "morse".

**13. Make a multilevel dictionary called life. Use these strings for the topmost keys: ‘animals’, ‘plants’**

**and ‘other’. Make the ‘animals’ key refer to another dictionary with the keys ‘cats’, ‘octopi’, and**

**‘emus’. Make the ‘cats’ key refer to a list of strings with the values ‘Henri’, ‘Grumpy’, and ‘Lucy’.**

**Make all the other keys refer to empty dictionaries.**

Absolutely! Here's how you create that multilevel dictionary in Python:

Python

life = {

"animals": {

"cats": ["Henri", "Grumpy", "Lucy"],

"octopi": {},

"emus": {}

},

"plants": {},

"other": {}

}

**Explanation:**

* **life:** This is the name of the main dictionary.
* **Top-level Keys:** The keys 'animals', 'plants', and 'other' represent the broad categories within your "life" dictionary.
* **Nested Dictionary:** The 'animals' key holds another dictionary within it. This is where you have more specific animal categories.
* **List of Cats:** The 'cats' key points to a list containing the names 'Henri', 'Grumpy', and 'Lucy'.
* **Empty Dictionaries:** The 'octopi', 'emus', 'plants', and 'other' keys currently point to empty dictionaries. You can add more details to them later.

**Let's explore this structure:**

* life['animals'] would give you the nested dictionary: { "cats": ["Henri", "Grumpy", "Lucy"], "octopi": {}, "emus": {} }
* life['animals']['cats'] would give you the list: ["Henri", "Grumpy", "Lucy"]
* life['plants'] would give you an empty dictionary: {}

**Feel free to expand!**

You can easily add more information to this dictionary. For example:

Python

life["animals"]["octopi"]["species"] = "Giant Pacific Octopus"

life["plants"]["types"] = ["Flowering", "Trees", "Ferns"]

**15. Print the keys for life[‘animals’].**

Absolutely! Here's how to create that multilevel dictionary in Python:

Python

life = {

"animals": {

"cats": ["Henri", "Grumpy", "Lucy"],

"octopi": {},

"emus": {}

},

"plants": {},

"other": {}

}

print(life["animals"].keys())

**Explanation:**

1. **Nested Structure:**
   * The life dictionary is the main dictionary, with top-level keys "animals", "plants", and "other".
   * The value for the "animals" key is itself another dictionary, creating the nested (multilevel) structure.
2. **Inner Dictionary:**
   * Inside the "animals" dictionary, you have the keys "cats", "octopi", and "emus".
   * The "cats" key has a list of names as its value. The other keys point to empty dictionaries, allowing you to add more animal categories later if you like.
3. **Accessing and Printing:**
   * life["animals"] accesses the dictionary within the "animals" key.
   * .keys() gives you the keys of this inner dictionary.

**Output:**

The print statement will produce the following output:

dict\_keys(['cats', 'octopi', 'emus'])

**16. Print the values for life[‘animals’][‘cats’]**

Absolutely! Here's how to create the multilevel dictionary and print the requested values:

**Python Code:**

Python

life = {

'animals': {

'cats': ['Henri', 'Grumpy', 'Lucy'],

'octopi': {},

'emus': {}

},

'plants': {},

'other': {}

}

# Print the keys for life['animals']

print(life['animals'].keys())

# Print the values for life['animals']['cats']

print(life['animals']['cats'])

**Explanation:**

1. **Creating the Structure:**
   * We start with a dictionary named life. It has three top-level keys: 'animals', 'plants', and 'other'.
   * The value for 'animals' is another dictionary (this is where it becomes multilevel).
   * Inside the 'animals' dictionary, we have keys 'cats', 'octopi', and 'emus'.
   * The value for 'cats' is a list containing the names 'Henri', 'Grumpy', and 'Lucy'.
   * The values for 'octopi' and 'emus' are empty dictionaries.
2. **Printing Keys and Values**
   * print(life['animals'].keys()) will print the keys within the 'animals' dictionary:
   * dict\_keys(['cats', 'octopi', 'emus'])
   * print(life['animals']['cats']) will print the list of cat names:
   * ['Henri', 'Grumpy', 'Lucy']