<제1장 신기한 보물 상자, 딕셔너리를 만나다!>

선생님: 얘들아, 안녕! 오늘 우리는 파이썬 나라의 아주 특별한 보물 상자를 열어볼 거란다. 그 이름은 바로 **딕셔너리(Dictionary)**! 🔑🎁

코딩새싹 🌱: 딕셔너리요? 혹시… 우리가 단어 뜻 찾을 때 쓰는 그 두꺼운 사전 말씀이세요? 📚

선생님: 오! 예리한데, 새싹아? 맞아! 그 사전이랑 아주 비슷해! 사전에서 '사과'라는 단어를 찾으면 '빨갛고 맛있는 과일'이라는 뜻이 나오지? 딕셔너리도 똑같아. '이름표'에 해당하는 **키(key)**를 주면, 그 이름표가 붙어있는 물건, 즉 **값(value)**을 쏙! 하고 알려준단다.

궁금이 💡: 이름표(키)랑 물건(값)이요? 음… 그럼 열쇠(키)로 열면 보물(값)이 나오는 상자 같아요! 🗝️➡️💎

선생님: (무릎을 탁 치며) 바로 그거야, 궁금아! 키로 값을 찾는 것! 마치 열쇠로 보물 상자를 여는 것과 같지! 딕셔너리는 이 **'키-값' 쌍**으로 이루어진 멋진 자료구조란다. 중요한 건, 각 키는 딱 하나만 있어야 해. 같은 이름표를 가진 물건이 여러 개면 헷갈리겠지?

코딩새싹 🌱: 아하! 이름표는 딱 하나! 그래서 그 이름표만 알면 바로 물건을 찾을 수 있군요!

선생님: 그렇지! 그리고 이 딕셔너리라는 보물 상자는 아주 유연해서, 우리가 원할 때마다 물건을 넣거나(추가), 바꾸거나(수정), 빼낼(삭제) 수도 있단다. 이걸 어려운 말로 '가변(mutable) 자료형'이라고 해. 하지만 지금은 '마음대로 바꿀 수 있는 신기한 상자' 정도로 기억해도 좋아! 또 하나 재미있는 건, 상자 안에 물건을 넣은 순서대로 꺼내지는 않는다는 거야. 이름표(키)로만 물건을 찾는 거지! (최신 파이썬에서는 넣은 순서를 기억하기도 하지만, 기본적으로는 이름표로 찾는 게 중요해!)

<제2장. 나만의 딕셔너리 보물 상자 만들기!>

선생님: 자, 그럼 우리도 이 멋진 딕셔너리 보물 상자를 직접 만들어 볼까? 먼저, 텅 빈 상자를 하나 만들어 보자!

```
# 빈 딕셔너리 생성
my_dict = {}
print(my_dict)
```

코딩새싹 🌱: 와! {} 이렇게 중괄호만 쓰면 텅 빈 상자가 뚝딱 만들어지네요! 신기해요! 그럼 이제 여기에 보물을 넣어볼까요?

선생님: 물론이지! 이번에는 처음부터 보물을 넣어서 상자를 만들어 보자. 예를 들어, 우리들의 친구 'John'에 대한 정보를 담아볼까?

```
# 키-값 쌍을 사용하여 딕셔너리 생성
person = {"name": "John", "age": 30, "city": "New York"}
print(person)
```

궁금이 💡: 오! "name"이 키고, "John"이 값인 건가요? 그럼 "age"라는 키에는 숫자 30이 값으로 들어있는 거고요?

선생님: 정확해, 궁금아! "name", "age", "city" 같은 것들이 바로 이름표, 즉 '키'가 되는 거고, 그 뒤에 콜론 :을 찍고 "John", 30, "New York" 같은 실제 정보, 즉 '값'을 적어주는 거란다. 키와 값은 따옴표로 감싸 문자열로 만들 수도 있고, 숫자 그대로 쓸 수도 있어!

<제3장. 보물 상자 속 내용물을 자유자재로!>

선생님: 자, 이제 John의 정보가 담긴 person이라는 딕셔너리 상자가 생겼어. 그럼 John의 이름을 꺼내보고 싶으면 어떻게 할까?

코딩새싹 🌱: 음… 이름표가 "name"이니까… "name" 키를 사용하면 될까요?

선생님: 맞아! 이렇게 대괄호 [] 안에 키를 쏙 넣어주면 된단다.

```
# 키를 사용하여 값 접근
print(person["name"])  # 출력: John
```

궁금이 💡: 와! 진짜 "John"이 나왔어요! 그럼 만약에 John이 한 살 더 먹으면 나이 정보도 바꿀 수 있나요?

선생님: 물론이지! 딕셔너리는 가변 자료형이라고 했지? 얼마든지 바꿀 수 있어. John의 나이를 31살로 바꿔볼까?

```
# 키-값 쌍 수정
person["age"] = 31
print(person)  # 출력: {"name": "John", "age": 31, "city": "New York"}
```

코딩새싹 🌱: 우와! "age" 키에 연결된 값이 30에서 31로 바뀌었어요! 그럼 새로운 정보도 추가할 수 있어요? 예를 들면 John의 직업 같은 거요!

선생님: 아주 좋은 질문이야! 당연히 추가할 수 있지. 새로운 키와 값을 정해서 쏙 넣어주기만 하면 돼.

```
# 새로운 키-값 쌍 추가
person["job"] = "Engineer"
```

```
print(person)  # 출력: {"name": "John", "age": 31, "city": "New York", "job": "Engineer"}
```

궁금이 💡: 와! "job"이라는 새로운 이름표랑 "Engineer"라는 값이 추가됐어요! 딕셔너리, 정말 신기한 상자인데요?

<제4장. 딕셔너리 보물 상자의 특별한 마법 주문! (메서드)>

선생님: 딕셔너리에는 아주 유용한 마법 주문, 즉 '메서드'라는 것들이 있단다. 이걸 사용하면 딕셔너리 상자를 더 편리하게 다룰 수 있어!

코딩새싹 🌱: 마법 주문이요? 어떤 게 있는데요? 😮

선생님: 첫 번째 마법 주문은 keys()! 이 주문을 외치면 상자 안에 있는 모든 이름표(키)들을 보여줘!

```
# keys() 메서드: 딕셔너리의 모든 키를 반환
print(person.keys())
```

궁금이 💡: dict_keys(['name', 'age', 'city', 'job']) 이렇게 나왔어요! 정말 이름표들만 쏙 보여주네요!

선생님: 그렇지? 다음은 values() 주문! 이건 뭘까요?

코딩새싹 🌱: 음… keys()가 이름표를 보여줬으니까… values()는 물건(값)들을 보여주겠네요!

선생님: 정답! 아주 똑똑한걸!

```
# values() 메서드: 딕셔너리의 모든 값을 반환
print(person.values())
```

궁금이 💡: dict_values(['John', 31, 'New York', 'Engineer']) 와! 값들만 쏙!

선생님: 세 번째 마법 주문은 items()! 이건 이름표와 물건을 예쁘게 짝지어서 한꺼번에 보여준단다.

```
# items() 메서드: 딕셔너리의 모든 키-값 쌍을 튜플로 반환
print(person.items())
```

코딩새싹 🌱: dict_items([('name', 'John'), ('age', 31), ('city', 'New York'), ('job', 'Engineer')]) 진짜 이름표랑 물건이 같이 묶여서 나왔어요! 😲

선생님: 마지막으로 아주 유용한 마법 주문, get()을 알려줄게. 이건 키를 사용해서 값을 가져오는 건 똑같은데, 만약 상자 안에 없는 이름표를 달라고 하면 어떻게 될까? person["없는키"] 이렇게 하면 "그런 키는 없어!"

하고 프로그램이 화를 내면서 멈출 수도 있거든.

궁금이 💡: 앗, 그럼 안 되잖아요! 😱

선생님: 그래서 get() 주문이 있는 거야! get()을 사용하면, 만약 찾는 키가 없더라도 화내지 않고 "음… 그건 없는데?" 하고 얌전히 알려주거나 (None), 우리가 미리 정해둔 "이거라도 대신 가져가렴~" 하는 기본값을 돌려준단다.

```
# get() 메서드: 키에 해당하는 값을 반환, 키가 없으면 None 또는 지정한 기본값 반환
print(person.get("age"))      # 있는 키를 찾으면: 30 (처음 값 기준) 또는 31 (수정된 값 기준)
print(person.get("country"))  # 없는 키를 찾으면? -> None (아무것도 없다는 뜻)
print(person.get("country", "Unknown")) # 없는 키를 찾을 때, "Unknown"이라고 대신 알려줘!
```

코딩새싹 🌱: 와, get() 정말 착한 마법 주문이네요! 프로그램이 갑자기 멈추는 걸 막아줄 수 있겠어요!

<제5장. 보물 상자 속 보물들을 하나씩 꺼내보자! (순회)>

선생님: 자, 이제 우리 딕셔너리 보물 상자 안에 어떤 보물들이 있는지 하나씩 꺼내보면서 구경해 볼 시간이야! 이걸 '순회한다'고 말한단다. 마치 박물관을 한 바퀴 돌면서 전시품들을 구경하는 것과 같아.

궁금이 💡: 와! 어떻게 구경할 수 있는데요?

선생님: 여러 가지 방법이 있어! 먼저, 이름표(키)들만 쭉~ 살펴볼 수 있지. for라는 마법 지팡이를 사용해서 말이야!

```
# 딕셔너리의 키 순회:
print("--- 키(이름표) 목록 ---")
for key in person:
    print(key)
```

코딩새싹 🌱: name, age, city, job 이렇게 이름표들이 차례대로 나왔어요!

선생님: 그렇지? 그럼 이번에는 이름표랑 그 이름표에 해당하는 물건(값)을 같이 보고 싶다면?

```
# 딕셔너리의 값 순회 (키를 이용해서):
print("--- 이름표와 물건 함께 보기 (방법1) ---")
for key in person:
    print(key + " 이름표에는 이런 물건이! -> " + str(person[key])) # 숫자는 str()로 글자로 바꿔줘야 같이
```

궁금이 💡: 와! "name 이름표에는 이런 물건이! -> John" 이렇게 나오네요! person[key]를 써서 물건을 꺼냈군요!

선생님: 똑똑해! 아까 배운 values() 마법 주문을 써서 물건(값)들만 쏙쏙 골라볼 수도 있어.

```
# 딕셔너리의 값 순회 (values() 사용):
print("--- 물건(값) 목록 ---")
for value in person.values():
    print(value)
```

코딩새싹 🌱: John, 31, New York, Engineer! 진짜 물건들만 쏙!

선생님: 마지막으로, 아까 items() 마법 주문 기억나니? 이름표랑 물건을 짝지어서 보여주는 거! 이걸 for 지팡이랑 같이 쓰면 정말 편하게 둘 다 꺼내볼 수 있단다.

```
# 딕셔너리의 키-값 쌍 순회:
print("--- 이름표와 물건 짝지어 보기 (items() 사용) ---")
for key, value in person.items():
    print(key + ": " + str(value)) # 이번에도 숫자는 str()!
```

궁금이 💡: name: John, age: 31 이렇게 깔끔하게 짝지어서 다 보여주네요! items()랑 for를 같이 쓰니 정말 편리해요!

선생님: (흐뭇하게 웃으며) 그래, 아주 잘 따라와 줬구나! 오늘 우리는 키와 값으로 이루어진 신기한 보물 상자, 딕셔너리에 대해 배웠어. 정보를 이름표와 함께 저장하고, 찾고, 바꾸고, 또 다양한 마법 주문(메서드)으로 편리하게 다루는 방법까지! 어때, 딕셔너리랑 조금 친해진 것 같니?

코딩새싹 🌱: 네, 선생님! 처음엔 어려울 것 같았는데, 비유로 설명해주시니까 귀에 쏙쏙 들어왔어요! 이제 저만의 보물 상자를 만들어서 이것저것 넣어보고 싶어요!

궁금이 💡: 저도요! 궁금한 게 생기면 또 질문해도 되죠? 딕셔너리로 게임 캐릭터 정보나, 좋아하는 동물들 정보를 정리해봐도 재미있을 것 같아요!

선생님: 물론이지! 언제든 환영이란다! 코딩은 이렇게 상상하고 직접 만들어보는 과정에서 더욱 즐거워지는 법이거든! 오늘 정말 수고 많았다, 우리 똑똑이 탐험가들! 👍