allinpython.com
@allinpython

Save This Post
For Future

Python Dictionary Methods

`.keys()`: Returns a list of all the keys in the dictionary.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
keys = my_dict.keys()
print(keys)  # Output: dict_keys(['name', 'age', 'gender'])
```

`.values()`: Returns a list of all the values in the dictionary.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
values = my_dict.values()
print(values)  # Output: dict_values(['John', 30, 'male'])
```

allinpython.com
@allinpython

Save This Post
For Future

.get(key, default): Retrieves the value associated with a given key. If the key is not found, it returns the specified default value.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
name = my_dict.get('name', 'Unknown')
print(name)  # Output: John

email = my_dict.get('email', 'Not specified')
print(email)  # Output: Not specified
```

.items(): Returns a list of key-value pairs as tuples.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
items = my_dict.items()
print(items)
# Output:
# dict_items([('name', 'John'), ('age', 30), ('gender', 'male')])
```

allinpython.com
@allinpython

Save This Post
For Future

.update(other_dict): Updates the dictionary with key-value pairs from another dictionary or an iterable.

```
1  my_dict = {'name': 'John', 'age': 30}
2  new_data = {'gender': 'male'}
3  my_dict.update(new_data)
4  print(my_dict)
5  # Output: {'name': 'John', 'age': 30, 'gender': 'male'}
```

.clear(): Removes all the key-value pairs from the dictionary, making it empty.

```
1  my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
2  my_dict.clear()
3  print(my_dict)  # Output: {}
```

allinpython.com
@allinpython

Save This Post
For Future

.pop(key[, default]): Removes and returns the value associated with the given key. If the key is not found, it returns the specified default value or raises a KeyError.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
age = my_dict.pop('age')
print(age)  # Output: 30

email = my_dict.pop('email', 'Not specified')
print(email)  # Output: Not specified
```

.copy(): Returns a shallow copy of the dictionary.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
copy_dict = my_dict.copy()
print(copy_dict)
# Output: {'name': 'John', 'age': 30, 'gender': 'male'}
```

allinpython.com
@allinpython

Save This Post
For Future

.popitem(): Removes and returns a key-value pair as a tuple in LIFO (Last In, First Out) order. Raises a KeyError if the dictionary is empty.

```python
my_dict = {'name': 'John', 'age': 30, 'gender': 'male'}
item = my_dict.popitem()
print(item)  # Output: ('gender', 'male')
print(my_dict)  # Output: {'name': 'John', 'age': 30}
```

.fromkeys(seq[, value]): Creates a new dictionary with keys from the given sequence (e.g., list, tuple) and values set to the specified value (defaulting to None if not provided).

```python
keys = ['name', 'age', 'gender']
values = 'Unknown'
my_dict = dict.fromkeys(keys, values)
print(my_dict)
# Output:
# {'name': 'Unknown', 'age': 'Unknown', 'gender': 'Unknown'}
```

allinpython.com
@allinpython

Save This Post
For Future

`.setdefault(key[, default])`: Returns the value associated with the given key.

If the key is not found, it inserts the key-value pair into the dictionary with the specified default value (defaulting to None if not provided).

```python
my_dict = {'name': 'John', 'age': 30}
name = my_dict.setdefault('name', 'Unknown')
print(name)  # Output: John

email = my_dict.setdefault('email', 'Not specified')
print(email)  # Output: Not specified
print(my_dict)
# Output: {'name': 'John', 'age': 30, 'email': 'Not specified'}
```

allinpython.com
@allinpython

Save This Post
For Future

For more Notes and Ebooks visit our

allinpython