Python Dictionaries Exam - Answers by Daud

Student Name: Daud

Date: 2025-07-29

Part 1: Multiple Choice Questions (MCQs) - Answers

1. (c)
2. (c)
3. (a) [Note: Raises KeyError, not generic 'Error']
4. (b) [Note: It overwrites the key's value]
5. (d) [Note: Tuples are valid keys only if they contain immutable types]
6. (a) [Note: Question context was incomplete]
7. (b)
8. (b)
9. (a)
10. (b)
11. (c)
12. (b)

Part 2: Structured Questions - Answers

13. A dictionary is a built-in data type that stores data in key-value pairs using {}. It is ordered (3.7+), changeable, unindexed, and does not allow duplicate keys.
14. dict.get(key) returns None or default value if the key doesn't exist; dict[key] raises KeyError.
15. Expression: if "name" in thisdict:
16. car["year"] = 2020 or car.update({"year": 2020})
17. Using dict.copy() or dict() constructor.
18. {'a': 0, 'b': 0}
19. [Incorrect] Correct answer: dict1 | dict2 (Python 3.9+)
20. Keys must be unique and immutable; accessing a missing key raises KeyError.
21. {'y': 2}
22. setdefault() returns value if key exists; sets and returns default if not.

Example:

```
d = {"x": 1, "y": 2}
mydict = d.setdefault("x", 1)
print(mydict)  # Output: 1
```

Part 3: Real Coding Questions - Answers and Feedback

23.

```
student = {"name" : "john", "age" : 25, "grade": "A"}
print(student)
```

24. Function not fully complete or logical for the task. A correct version:

```
def get_numeric_keys(d):
    return [k for k, v in d.items() if isinstance(v, (int, float))]
```

25.

```
keys = ["name", "age", "gender"]
values = ["Ali", 25, "Male"]
mydict = dict(zip(keys, values))
```

26. Acceptable but not dictionary-based. Recommended:

```
counter = {}
for word in words:
    counter[word] = counter.get(word, 0) + 1
```

27. Long but correct. Shorter version:

```
for k in mydict:
    mydict[k] *= 1.10
```

28. Correct solution:

```
def swap_dict(d):
```

```
    return {v: k for k, v in d.items()}
```

29. Correct solution:

```
max_key = max(data, key=data.get)
```

30. Correct solution:

```
def extract_names(nested):
    return [info["name"] for info in nested.values()]
```