1. **Cú pháp nào đúng để khai báo một biến trong Python?**
    - a) `int x = 5`
    - b) `x = 5`
    - c) `var x = 5`
    - d) `dim x = 5`

2. **Câu lệnh nào dùng để gán giá trị `10` cho biến `x` và `y` cùng lúc?**
    - a) `x = y = 10`
    - b) `x = 10; y = 10`
    - c) `x == y == 10`
    - d) `x = y: 10`

3. **Python là một ngôn ngữ kiểu gì?**
    - a) Kiểu tĩnh
    - b) Kiểu động
    - c) Kiểu trung bình
    - d) Không có kiểu

4. **Câu lệnh nào sẽ chuyển đổi chuỗi `"123"` thành số nguyên?**
    - a) `int("123")`
    - b) `str(123)`
    - c) `float("123")`
    - d) `num("123")`

5. **Kết quả của biểu thức `type(5.0)` là gì?**
    - a) `<class 'int'>`
    - b) `<class 'str'>`
    - c) `<class 'float'>`
    - d) `<class 'complex'>`

6. **Phát biểu nào sau đây là đúng về biến trong Python?**
    - a) Biến cần khai báo trước khi sử dụng
    - b) Biến có thể thay đổi kiểu dữ liệu bất kỳ lúc nào
    - c) Biến phải là số nguyên
    - d) Biến không thể được gán lại

7. **Câu lệnh nào sẽ trả về kiểu dữ liệu của biến `x`?**
    - a) `type(x)`
    - b) `typeof(x)`
    - c) `class(x)`
    - d) `dtype(x)`
8. **Phát biểu nào sau đây là sai về biến toàn cục?**
    - a) Biến toàn cục có thể được truy cập từ mọi nơi trong chương trình
    - b) Biến toàn cục phải được khai báo ở đầu chương trình
    - c) Biến toàn cục tồn tại cho đến khi chương trình kết thúc
    - d) Biến toàn cục có thể được sửa đổi bên trong hàm sử dụng từ khóa `global`
9. **Câu lệnh nào sẽ tạo ra lỗi khi chạy trong Python?**
    - a) `x = 10`
    - b) `y = 3.14`
    - c) `z = 'Hello'`
    - d) `print(x + z)`
10. **Cú pháp nào dùng để khai báo hằng số trong Python?**
    - a) `const PI = 3.14`
    - b) `PI = 3.14`
    - c) `PI := 3.14`
    - d) `hằng số không thể được khai báo trong Python`
11. **Cú pháp nào dùng để tạo vòng lặp `for` trong Python?**
    - a) `for i in range(10):`
    - b) `for (i = 0; i < 10; i++):`
    - c) `for i to 10:`
    - d) `for i in 10:`
12. **Câu lệnh `while` trong Python kết thúc khi nào?**
    - a) Khi điều kiện là `True`
    - b) Khi điều kiện là `False`
    - c) Khi gặp lệnh `break`
    - d) Cả b và c

13. **Kết quả của đoạn mã sau là gì?**

```
i = 0
while i < 3:
    print(i)
    i += 1
```

- a) 0 1 2
- b) 1 2 3
- c) 0 1 2 3
- d) 0 1

14. **Lệnh `continue` trong vòng lặp có chức năng gì?**

- a) Kết thúc vòng lặp
- b) Bỏ qua phần còn lại của vòng lặp và chuyển đến lần lặp tiếp theo
- c) Trả về giá trị trong vòng lặp
- d) In ra giá trị hiện tại

15. **Lệnh nào dưới đây kết thúc vòng lặp ngay lập tức?**

- a) `continue`
- b) `pass`
- c) `break`
- d) `return`

16. **Câu lệnh `for i in range(5)` sẽ lặp bao nhiêu lần?**

- a) 4 lần
- b) 5 lần
- c) 6 lần
- d) 3 lần

17. **Phát biểu nào sau đây là đúng về vòng lặp lồng nhau trong Python?**

- a) Chỉ vòng lặp `for` mới có thể lồng nhau
- b) Bạn có thể lồng bất kỳ loại vòng lặp nào
- c) Vòng lặp không thể lồng nhau
- d) Vòng lặp chỉ có thể lồng tối đa 2 lần

18. **Kết quả của đoạn mã sau là gì?**

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

- a) 0 1 2 3 4 5
- b) 0 0 0 1 1 0 1 1
- c) 0 0 0 1 1 0 1 1 2 0 2 1
- d) 0 1 0 1 0 1

19. **Cú pháp đúng để tạo một vòng lặp `for` với danh sách `numbers` là gì?**

- a) `for number in numbers:`
- b) `for i = 0; i < len(numbers); i++:`
- c) `for i to numbers:`
- d) `for numbers in number:`

20. **Kết quả của đoạn mã sau là gì?**

```
for i in range(3):
    if i == 1:
        break
    print(i)
```

- a) 0 1
- b) 0
- c) 0 1 2
- d) Không có gì được in ra

21. **Câu lệnh nào dùng để kiểm tra điều kiện trong Python?**

- a) `if`
- b) `switch`
- c) `case`
- d) `unless`
- **Đáp án: a)**

22. **Kết quả của đoạn mã sau là gì?**

```
x = 10
if x > 5:
```

```
    print("Lớn hơn 5")
else:
    print("Nhỏ hơn hoặc bằng 5")
```

- a) Lớn hơn 5
- b) Nhỏ hơn hoặc bằng 5
- c) Không có gì được in ra
- d) Lỗi

23. **Câu lệnh `elif` được sử dụng để làm gì?**

- a) Kết thúc một câu lệnh `if`
- b) Bắt đầu một câu lệnh điều kiện mới
- c) Kiểm tra điều kiện khác nếu điều kiện ban đầu là `False`
- d) Định nghĩa một hàm

24. **Kết quả của đoạn mã sau là gì?**

```
x = 10
if x > 15:
    print("Lớn hơn 15")
elif x > 5:
    print("Lớn hơn 5")
else:
    print("Nhỏ hơn hoặc bằng 5")
```

- a) Lớn hơn 15
- b) Lớn hơn 5
- c) Nhỏ hơn hoặc bằng 5
- d) Không có gì được in ra

25. **Câu lệnh nào để kiểm tra tính đúng đắn của nhiều điều kiện?**

- a) `if`
- b) `else`
- c) `elif`
- d) `if-else`

26. **Cú pháp đúng để kiểm tra nếu `x` không bằng `y` là gì?**

- a) `if x != y:`

- b) `if x <> y:`
- c) `if x =! y:`
- d) `if x not y:`

27. **Phát biểu nào sau đây là đúng về câu lệnh `if-else`?**

- a) `else` luôn đi kèm với `if`
- b) `else` là bắt buộc sau mỗi `if`
- c) `else` có thể đi trước `if`
- d) `else` có thể đi kèm với `elif`

28. **Kết quả của đoạn mã sau là gì?**

```
x = 10
y = 20
if x > y:
    print("x lớn hơn y")
else:
    print("x không lớn hơn y")
```

- a) x lớn hơn y
- b) x không lớn hơn y
- c) Không có gì được in ra
- d) Lỗi

29. **Câu lệnh nào kiểm tra nếu một biến `x` có giá trị dương?**

- a) `if x > 0:`
- b) `if x == 0:`
- c) `if x >= 0:`
- d) `if x != 0:`

30. **Cú pháp đúng để kiểm tra nhiều điều kiện cùng một lúc là gì?**

- a) `if x > 0 and y > 0:`
- b) `if x > 0 or y > 0:`
- c) `if x > 0 & y > 0:`
- d) Cả a và b

31. **Danh sách (List) trong Python là gì?**

- a) Một tập hợp không có thứ tự

- b) Một tập hợp có thể thay đổi và có thứ tự
- c) Một tập hợp không thể thay đổi
- d) Một tập hợp có thể thay đổi nhưng không có thứ tự

32. **Cú pháp đúng để tạo một danh sách trong Python là gì?**

- a) `list = (1, 2, 3)`
- b) `list = [1, 2, 3]`
- c) `list = {1, 2, 3}`
- d) `list = <1, 2, 3>`

33. **Cú pháp nào để thêm một phần tử vào danh sách `my_list`?**

- a) `my_list.add(10)`
- b) `my_list.append(10)`
- c) `my_list.insert(10)`
- d) `my_list.extend(10)`

34. **Kết quả của lệnh `len([1, 2, 3, 4])` là gì?**

- a) 3
- b) 4
- c) 5
- d) 2

35. **Lệnh nào sẽ xóa phần tử có giá trị `5` trong danh sách `my_list`?**

- a) `my_list.remove(5)`
- b) `my_list.pop(5)`
- c) `del my_list[5]`
- d) `my_list.delete(5)`

36. **Kết quả của đoạn mã sau là gì?**

```
my_list = [1, 2, 3]
my_list.extend([4, 5])
print(my_list)
```

- a) `[1, 2, 3, 4, 5]`
- b) `[1, 2, 3, [4, 5]]`
- c) `[4, 5, 1, 2, 3]`

- d) `[1, 2, 3]`

37. **Lệnh nào sẽ trả về phần tử cuối cùng trong danh sách `my_list`?**

    - a) `my_list[0]`
    - b) `my_list[-1]`
    - c) `my_list[1]`
    - d) `my_list[1:]`

38. **Danh sách trong Python có thể chứa các phần tử thuộc các kiểu dữ liệu khác nhau không?**

    - a) Có
    - b) Không

39. **Kết quả của đoạn mã sau là gì?**

```
my_list = [1, 2, 3, 4]
my_list[1:3] = [10, 20]
print(my_list)
```

    - a) `[1, 10, 20, 4]`
    - b) `[1, 2, 3, 10, 20, 4]`
    - c) `[1, 10, 3, 4]`
    - d) `[1, 2, 3, 4]`

40. **Lệnh nào để tạo ra một danh sách mới từ danh sách `my_list` với các phần tử được sắp xếp theo thứ tự tăng dần?**

    - a) `sorted_list = sorted(my_list)`
    - b) `my_list.sort()`
    - c) `sorted_list = my_list.sort()`
    - d) `my_list = sorted_list(my_list)`

41. **Tuple là gì trong Python?**

    - a) Một loại danh sách có thể thay đổi
    - b) Một danh sách không thể thay đổi
    - c) Một danh sách các từ điển
    - d) Một danh sách các số nguyên

42. **Cú pháp đúng để tạo một tuple trong Python là gì?**

- a) `tuple = [1, 2, 3]`
- b) `tuple = {1, 2, 3}`
- c) `tuple = (1, 2, 3)`
- d) `tuple = <1, 2, 3>`

43. **Cú pháp nào đúng để truy cập phần tử thứ hai trong một tuple?**

- a) `tuple[1]`
- b) `tuple(2)`
- c) `tuple[2]`
- d) `tuple.get(1)`

44. **Tuple có thể thay đổi không?**

- a) Có
- b) Không

45. **Cú pháp nào sẽ tạo ra một tuple rỗng?**

- a) `empty_tuple = ()`
- b) `empty_tuple = []`
- c) `empty_tuple = {}`
- d) `empty_tuple = None`

46. **Lệnh nào dưới đây sẽ dẫn đến lỗi khi thực hiện với một tuple?**

- a) `len(tuple)`
- b) `tuple[0]`
- c) `tuple[1] = 10`
- d) `tuple.index(2)`

47. **Làm cách nào để nối hai tuple trong Python?**

- a) `tuple1 + tuple2`
- b) `tuple1 - tuple2`
- c) `tuple1 * tuple2`
- d) `tuple1 / tuple2`

48. **Cách nào để chuyển đổi một danh sách thành tuple?**

- a) `tuple(list)`
- b) `list(tuple)`
- c) `set(list)`
- d) `dict(list)`

49. **Tuple có thể chứa các phần tử thuộc các kiểu dữ liệu khác nhau không?**

    - a) Có
    - b) Không

50. **Kết quả của lệnh `tuple1[0:2]`**

là gì nếu `tuple1 = (10, 20, 30, 40)`? - a) `(10, 20, 30)` - b) `(20, 30)` - c) `(10, 20)` - d) `(10, 30)`

51. **Dictionary trong Python là gì?**

    - a) Một danh sách các cặp key-value
    - b) Một tập hợp các số nguyên
    - c) Một danh sách không có thứ tự
    - d) Một danh sách không thể thay đổi

52. **Cú pháp đúng để tạo một dictionary trong Python là gì?**

    - a) `dict = [1, 2, 3]`
    - b) `dict = {1, 2, 3}`
    - c) `dict = {"a": 1, "b": 2}`
    - d) `dict = (1, 2, 3)`

53. **Lệnh nào sẽ thêm một cặp key-value vào dictionary `dict1`?**

    - a) `dict1.append("a", 1)`
    - b) `dict1["a"] = 1`
    - c) `dict1.add("a", 1)`
    - d) `dict1["a", 1]`

54. **Lệnh `dict1.get("a")` trả về điều gì nếu key `"a"` không tồn tại trong dictionary?**

    - a) `0`
    - b) `None`
    - c) `KeyError`

- d) `False`

55. **Làm cách nào để xóa một cặp key-value trong dictionary?**

    - a) `del dict1["key"]`
    - b) `remove dict1["key"]`
    - c) `dict1.pop("key")`
    - d) Cả a và c

56. **Lệnh `dict1.keys()` trả về điều gì?**

    - a) Tất cả các giá trị trong dictionary
    - b) Tất cả các key trong dictionary
    - c) Cả key và value trong dictionary
    - d) Số lượng các phần tử trong dictionary

57. **Cú pháp nào để kiểm tra xem một key có tồn tại trong dictionary không?**

    - a) `"key" in dict1`
    - b) `dict1.has("key")`
    - c) `dict1.exists("key")`
    - d) `"key" is dict1`

58. **Dictionary có hỗ trợ việc lặp qua các phần tử không?**

    - a) Có
    - b) Không

59. **Cú pháp nào đúng để xóa tất cả các phần tử trong một dictionary?**

    - a) `dict1.clear()`
    - b) `dict1.remove_all()`
    - c) `del dict1`
    - d) `dict1.delete()`

60. **Lệnh nào sẽ trả về số lượng phần tử trong một dictionary?**

    - a) `size(dict1)`
    - b) `len(dict1)`
    - c) `dict1.length()`
    - d) `count(dict1)`