PYTHON TEST - 1.12 (JUMP STATEMENTS)

Total points 50/50

Jump Statements in Python

STUDENT NAME *

VIVA

✓ 1. Which jump statement is used to exit a loop immediately in Python? * 1/1

- ○ a) exit
- ◉ b) break ✓
- ○ c) continue
- ○ d) stop

✓ 2. Which jump statement skips the current iteration and moves to the next in Python? *1/1

- a) break
- b) exit
- c) continue ✓
- d) stop

✓ 3. The break statement is used in: * 1/1

- a) if condition only
- b) loops only
- c) both loops and if condition ✓
- d) none

✓ 4. The continue statement is used in: * 1/1

- a) loops only ✓
- b) functions only
- c) conditional blocks only
- d) both loops and functions

✓ 5. Which loop terminates completely on encountering break? * 1/1

- (●) a) Only the innermost loop ✓
- ( ) b) All loops together
- ( ) c) Outer loop always
- ( ) d) None

✓ 6. What happens if break is used inside a for loop? * 1/1

- ( ) a) Loop skips one iteration
- (●) b) Loop terminates immediately ✓
- ( ) c) Loop continues infinitely
- ( ) d) Error occurs

✓ 7. Code: *
for i in range(5):
if i == 3:
break
print(i)
Output? 1/1

- ( ) a) 0 1 2 3 4
- (●) b) 0 1 2 ✓
- ( ) c) 0 1 2 3
- ( ) d) Error

✓ 8.In nested loops, break affects: * 1/1

- a) All loops
- b) Only the inner loop ✓
- c) The outer loop
- d) None

✓ 9. Which of the following can terminate a loop immediately? * 1/1

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

✓ 10. Code: * 1/1

```
while True:
    print("Hello")
    break
```

Output?

- a) Infinite Hello
- b) Hello (once) ✓
- c) No output
- d) Error

✓ 11. Code: * 1/1

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

Output?

- a) 0 1 2 3 4
- b) 0 1 3 4 ✓
- c) 2 3 4
- d) Error

✓ 12. What happens if continue is used inside a loop? * 1/1

- a) Exits the loop completely
- b) Skips the current iteration ✓
- c) Restarts the loop
- d) Terminates the program

13. Code: *

1/1

```
x = 0

while x < 5:

    x += 1

    if x == 3:

        continue
print(x
```

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

14. Which statement is used to skip executing code inside a loop for a particular iteration? *

1/1

- a) break
- b) continue ✓
- c) exit
- d) stop

✓ 15. Code: * 1/1

```
for i in range(4):
    if i == 0:
        continue
    print(i)
```

Output?

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

✓ 16. Which statement stops loop execution completely? * 1/1

- a) break ✓
- b) continue
- c) exit()
- d) pass

✓ 17. Which statement skips remaining statements in the current iteration? * 1/1

- a) continue ✓
- b) break
- c) pass
- d) exit

✓ 18. Code: * 1/1

```
for i in range(5):
    if i < 3:
        continue
    else:
        break
    print(i)
```

Output?

- a) 3 4
- b) Infinite loop
- c) No output ✓
- d) Error

✓ 19. Which statement is more useful to skip certain values in loops? * 1/1

- a) break
- b) continue ✓
- c) exit()
- d) pass

✓ 20. Which statement is used to exit the loop but not the program? * 1/1

- a) continue
- b) break ✓
- c) exit()
- d) pass

✓ 21. Code: * 1/1

```
for ch in "python":
    if ch == "h":
        break
    print(ch)
```

Output?

- a) p y t
- b) p y t h
- c) p y ✓
- d) Error

✓ 22. Code: * 1/1

```
for ch in "python":
    if ch == "h":
        continue
    print(ch)
```

Output?

- a) python
- b) p y t o n ✓
- c) p y t h o n
- d) Error

✓ 23. Code: * 1/1

```
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)
```

Output?

- a) 0 2 4 6 8
- b) 1 3 5 7 9 ✓
- c) 2 4 6 8
- d) Error

✓ 24. Code: * 1/1

```
count = 0

while count < 5:

    count += 1

    if count == 4:

        break

    print(count)
```

Output?

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

✓ 25. Can break and continue be used outside loops? * 1/1

- a) Yes
- b) No ✓
- c) Only break
- d) Only continue

✓ 26. In nested loops, continue applies to: * 1/1

- ○ a) All loops
- ◉ b) Only the current loop where it is written ✓
- ○ c) Outer loop always
- ○ d) None

✓ 27. Code: *

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

Output? 1/1

- ○ a) (0,0)(0,1)(0,2)...
- ◉ b) (0,0)(1,0)(2,0) ✓
- ○ c) (0,0)(0,1)(1,0)...
- ○ d) None

✓ 28. Code: * 1/1

```
for i in range(3):
    for j in range(3):
        if j == 1:
            continue
        print(i, j)
```

Output?

- (●) a) All pairs except j=1 ✓
- ( ) b) Only j=1
- ( ) c) Infinite loop
- ( ) d) Error

✓ 29. Which is true? * 1/1

- ( ) a) break skips current iteration
- ( ) b) continue exits loop
- (●) c) break exits loop immediately ✓
- ( ) d) continue exits loop immediately

✓ 30. Can break and continue be used together in the same loop? * 1/1

- (●) a) Yes ✓
- ( ) b) No
- ( ) c) Only break allowed
- ( ) d) Only continue allowed

✓ 31. What will happen? * 1/1

```
for i in range(5):
    if i == 2:
        break
    else:
        continue
    print(i)
```

- ( ) a) 0 1
- ( ) b) 0 1 2
- (●) c) No output ✓
- ( ) d) Error

✓ 32. Code: * 1/1

```
for i in range(5):
    if i < 2:
        continue
    print(i)
```

Output?

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

✓ 33. Code: * 1/1

```
i = 0
while i < 5:
    i += 1
    if i == 5:
        break
    print(i)
```

Output?

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

✓ 34. The opposite of continue in loops is: * 1/1

- ○ a) pass
- ◉ b) break ✓
- ○ c) exit
- ○ d) return

✓ 35. What will be printed? * 1/1

```
for i in range(3):
    for j in range(3):
        if i == j:
            continue
        print(i, j)
```

Output?

- ◉ a) Skips equal pairs ✓
- ○ b) Prints all pairs
- ○ c) Prints only equal pairs
- ○ d) Error

✓ 36. Can continue stop a loop? * 1/1

- ○ a) Yes
- ◉ b) No ✓
- ○ c) Sometimes
- ○ d) Only in for loop

✓ 37. Can break be used in for loop with else? * 1/1

- (●) a) Yes ✓
- ( ) b) No
- ( ) c) Only in while
- ( ) d) Error

✓ 38. Code: * 1/1

```
for i in range(3):
    if i == 2:
        break
else:
    print("Completed") Output?
```

- ( ) a) Completed
- (●) b) No output ✓
- ( ) c) Error
- ( ) d) 0 1 2

✓ 39. Can continue affect loop else? * 1/1

- ( ) a) Yes
- (●) b) No ✓
- ( ) c) Only in while
- ( ) d) Error

✓ 40. break statement is generally used with: * 1/1

- a) switch case
- b) loops ✓
- c) functions
- d) classes

✓ 41. Code: *

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

Output? 1/1

- a) 0 Done
- b) 0 ✓
- c) 1
- d) Done

✓ 42. Code: * 1/1

```
for i in range(3):
    if i == 1:
        continue
    print(i)
else:
    print("Done")
```

Output?

- (●) a) 0 2 Done ✓
- ( ) b) 0 1 2 Done
- ( ) c) 0 Done
- ( ) d) Done only

✓ 43. Can break terminate infinite loop? * 1/1

- (●) a) Yes ✓
- ( ) b) No
- ( ) c) Only in while loop
- ( ) d) Only in for loop

✓ 44. Which of the following doesn't terminate loop execution? * 1/1

- a) break
- b) continue ✓
- c) return
- d) exit()

✓ 45. Can continue create infinite loop if not used carefully? * 1/1

- a) Yes ✓
- b) No

✓ 46. Which jump statement is optional in Python loops? * 1/1

- a) break
- b) continue
- c) both ✓
- d) None

✓ 47. break can also be used inside: * 1/1

- a) try-except
- b) functions
- c) switch-case (if implemented)
- d) None ✓

✓ 48. Code: *
1/1

```
while True:
    break
else:
    print("Else")
```

Output?

- a) Else
- b) No output ✓
- c) Infinite loop
- d) Error

✓ 49. Which keyword resumes next iteration of loop? * 1/1

- a) next
- b) continue ✓
- c) break
- d) skip

✓ 50. Which jump statement is not part of Python? * 1/1

- [ ] a) break
- [ ] b) continue
- [x] c) goto ✓
- [ ] d) None