

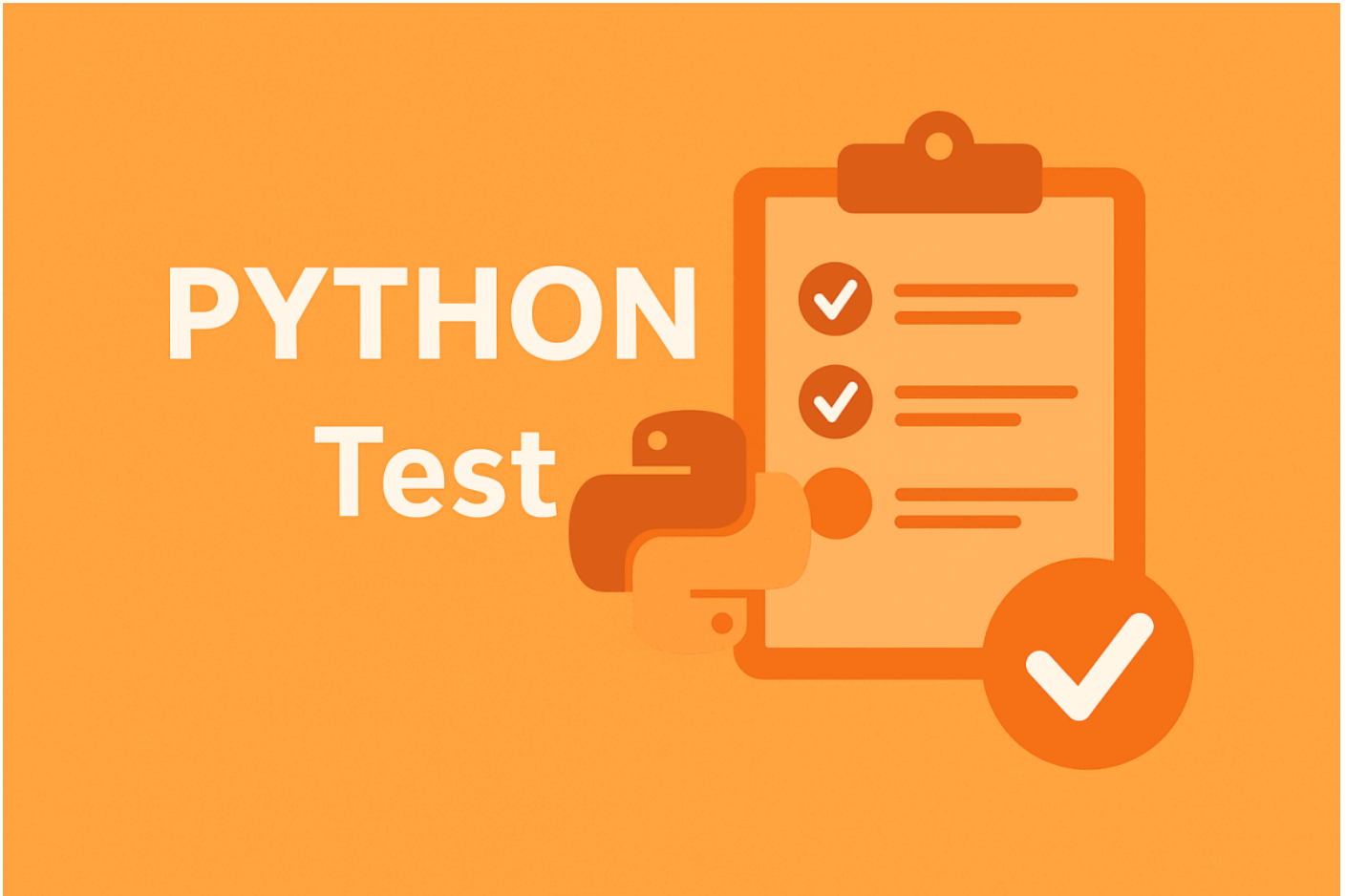
Geetest



Тест по основам
программирования
на Python

Тест по основам программирования на Python

Этот тест предназначен для проверки знаний и навыков работы с языком Python на базовом и среднем уровне.



Основы программирования на Python

Он охватывает 8 ключевых тем:

1. Циклы и управление потоками – `for` , `while` , `break` , `continue` , вложенные циклы, `else` в циклах.
2. Условные конструкции и логические выражения – `if` , `elif` , `else` , логические операторы.
3. Функции и модули – определение функций, аргументы, возвращаемые значения, использование встроенных и пользовательских модулей.
4. Переменные и типы данных – числа, строки, списки, кортежи, словари, множества, преобразования типов.
5. Структуры данных – списки, кортежи, словари, множества, работа с элементами, методы и операции.
6. Строки – методы строк, срезы, форматирование, f-строки, проверка содержимого и преобразования.
7. Файлы и работа с модулями – открытие, чтение, запись файлов, работа с директориями, динамический импорт модулей.
8. Исключения и обработка ошибок – `try/except/finally` , пользовательские исключения, `raise` , `assert` , обработка ошибок при работе с файлами и вычислениях.

Формат вопросов:

множественный выбор, с явным указанием верного ответа.

Тест подходит для:

- проверки знаний студентов и начинающих разработчиков
- подготовки к собеседованиям и внутренним тестированиям
- самостоятельного контроля уровня владения Python

Тема 1. Основы Python.

1. Какой тип данных будет у выражения `3 + 4.0`?

- 1) `int`
- 2)+ `float`
- 3) `str`
- 4) `bool`

2. Что выведет `print("Hello"[1])`?

- 1) `H`
- 2)+ `e`
- 3) `l`
- 4) `o`

3. Какой оператор используется для возведения в степень?

- 1) `%`
- 2) `//`
- 3)+ `**`
- 4) `^`

4. Как создать переменную `x` и присвоить ей значение 10?

- 1) `x == 10`
- 2)+ `x = 10`
- 3) `var x = 10`
- 4) `x := 10`

5. Что вернет `type(True)` ?

- 1) `str`
- 2) `int`
- 3)+ `bool`
- 4) `None`

6. Какой из этих литералов является строкой?

- 1) `123`
- 2)+ `"abc"`
- 3) `True`
- 4) `None`

7. Что выведет `print(7 // 2)` ?

- 1) `3.5`
- 2)+ `3`
- 3) `4`
- 4) `3.0`

8. Что выведет `print(7 % 3)` ?

- 1) `1.0`
- 2)+ `1`
- 3) `0`
- 4) `2`

9. Какой оператор сравнения проверяет равенство?

- 1) `=`
- 2)+ `==`
- 3) `===`
- 4) `!=`

10. Какой из вариантов создаёт булеву переменную со значением `False` ?

- 1) `x = 0`
- 2) `x = "False"`
- 3)+ `x = False`
- 4) `x = None`

11. Что делает функция `print()` ?

- 1)+ выводит данные на экран
- 2) сохраняет данные в файл
- 3) создает переменную
- 4) проверяет условие

12. Какой символ используется для комментария в Python?

- 1) `//`
- 2) `/* */`
- 3)+ `#`

13. Как правильно написать многострочный комментарий?

- 1) `/* comment */`
- 2)+ `""" comment """`
- 3) `// comment //`
- 4) `# comment #`

14. Что выведет `print(2 ** 3)` ?

- 1) 5
- 2) 6
- 3)+ 8
- 4) 9

15. Какой из типов данных неизменяемый?

- 1) `list`
- 2)+ `tuple`
- 3) `dict`
- 4) `set`

16. Что выведет `print(bool(""))` ?

- 1) `True`
- 2)+ `False`
- 3) `None`
- 4) 0

17. Что выведет `print(bool(1))` ?

- 1)+ `True`
- 2) `False`
- 3) `None`
- 4) `1`

18. Какой тип данных будет у `3 / 2` в Python 3?

- 1) `int`
- 2)+ `float`
- 3) `str`
- 4) `bool`

19. Что делает оператор `//` ?

- 1) деление с остатком
- 2)+ целочисленное деление
- 3) возведение в степень
- 4) деление по модулю

20. Что делает оператор `%` ?

- 1) целочисленное деление
- 2)+ остаток от деления
- 3) возведение в степень
- 4) проверка на делимость

21. Как создать пустой список?

- 1) `x = ()`
- 2) `x = {}`
- 3)+ `x = []`
- 4) `x = set()`

22. Как создать пустой кортеж?

- 1) `x = []`
- 2) `x = {}`
- 3)+ `x = ()`
- 4) `x = set()`

23. Как создать пустое множество?

- 1) `x = []`
- 2) `x = {}`
- 3) `x = ()`
- 4)+ `x = set()`

24. Как создать пустой словарь?

- 1) `x = []`
- 2) `x = ()`
- 3)+ `x = {}`
- 4) `x = set()`

25. Что выведет `print(len("Python"))` ?

- 1) 5
- 2)+ 6
- 3) 7
- 4) 0

26. Что делает функция `type()` ?

- 1)+ возвращает тип объекта
- 2) преобразует тип объекта
- 3) выводит объект на экран
- 4) проверяет условие

27. Как правильно объединить строки `a = "Hello"` и `b = "World"` ?

- 1) `a + b`
- 2)+ `a + b`
- 3) `a & b`
- 4) `a.concat(b)`

28. Какое значение присвоится переменной `x` после `x = 10; x += 5` ?

- 1) 10
- 2)+ 15
- 3) 5
- 4) 20

29. Что делает оператор `is` ?

- 1) проверяет равенство значений
- 2)+ проверяет, указывают ли переменные на один объект
- 3) присваивает значение
- 4) проверяет тип объекта

30. Как проверить, что переменная `x` принадлежит множеству `[1, 2, 3]` ?

- 1) `x in [1, 2, 3]`
- 2)+ `x in [1, 2, 3]`
- 3) `x == [1, 2, 3]`
- 4) `x = [1, 2, 3]`

31. Что выведет `print(3 > 2 and 2 > 1)` ?

- 1)+ `True`
- 2) `False`

32. Что выведет `print(3 > 2 or 2 < 1)` ?

- 1)+ `True`
- 2) `False`

33. Что выведет `print(not True)` ?

- 1)+ `False`
- 2) `True`

34. Какой тип данных у `None` ?

- 1) `int`
- 2) `str`
- 3) `bool`
- 4)+ `NoneType`

35. Что выведет `print(0.1 + 0.2 == 0.3)` ?

- 1) `True`
- 2)+ `False`
- 3) `Error`
- 4) `None`

Тема 2. Условные конструкции.

1. Что выведет код:

```
python
x = 5
if x > 3:
    print("A")
else:
    print("B")
```

- 1)+ A
- 2) B

2. Как правильно написать тернарный оператор, чтобы присвоить $y = 10$ если $x > 5$, иначе $y = 0$?

- 1) `y = x > 5 ? 10 : 0`
- 2)+ `y = 10 if x > 5 else 0`
- 3) `if x > 5: y = 10 else: y = 0`
- 4) `y = 0 if x > 5 else 10`

3. Что вернет `bool(0)` ?

- 1)+ False
- 2) True

4. В каком случае выполняется блок `elif`?

- 1) Всегда
- 2)+ Только если предыдущие `if` и `elif` не сработали
- 3) Только если `else` не выполняется
- 4) Ни при каких условиях

5. Какой из вариантов корректен для вложенного условия?

- 1) `if x > 5 if y < 3:`
- 2)+ `if x > 5: if y < 3:`
- 3) `if x > 5 else if y < 3:`
- 4) `if x > 5 elif y < 3:`

6. Что выведет код:

```
python
x = 10
if x < 5:
    print("A")
elif x < 15:
    print("B")
else:
    print("C")
```

- 1) A
- 2)+ B
- 3) C

7. Как проверить, что переменная x не равна 5?

- 1) `x != 5`
- 2)+ `x != 5`
- 3) `x <> 5`
- 4) `x =! 5`

8. Какой из вариантов синтаксически верен?

- 1) `if x > 0 then print(x)`
- 2)+ `if x > 0: print(x)`
- 3) `if x > 0 print(x)`
- 4) `if x > 0; print(x)`

9. Что делает блок else?

- 1) выполняется только если if верно
- 2)+ выполняется, если все предыдущие условия ложны
- 3) завершает программу
- 4) ничего

10. Какой из вариантов корректен для множественного условия?

- 1) `if x > 5 and y:`
- 2)+ `if x > 5 and y > 0:`
- 3) `if x > 5 & y > 0:`
- 4) `if x > 5 || y > 0:`

11. Что выведет код:

```
python
x = 7
if x % 2 == 0:
    print("even")
else:
    print("odd")
```

- 1) `even`
- 2)+ `odd`

12. Что выведет `print(True and False)` ?

- 1) `True`
- 2)+ `False`

13. Что выведет `print(True or False)` ?

- 1)+ `True`
- 2) `False`

14. Что делает оператор `not` ?

- 1)+ инвертирует логическое значение
- 2) проверяет равенство
- 3) присваивает значение
- 4) делит

15. Как проверить, что x больше 0 и меньше 10?

- 1) $0 < x < 10$
- 2)+ $0 < x < 10$
- 3) $x > 0 \text{ and } x < 10$
- 4) $x > 0 < 10$

16. Какой вариант верен для `if` с несколькими `elif` ?

- 1) `if x>0: ... elif x>1: ... elif x>2:`
- 2)+ `if x>2: ... elif x>1: ... elif x>0:`
- 3) `if x>2: ... else if x>1: ... else if x>0:`
- 4) `if x>2: ... else: if x>1: ... else: if x>0:`

17. Что выведет код:

```
python
x = 3
if x > 5:
    print("A")
else:
    print("B")
```

- 1) A
- 2)+ B

18. Как правильно записать условие для проверки, что x равно 10 или 20?

- 1) $x == 10 \ || \ x == 20$
- 2)+ $x == 10 \text{ or } x == 20$
- 3) $x = 10 \text{ or } 20$
- 4) $x \text{ in } [10] \text{ or } [20]$

19. Что делает оператор `and`?

- 1)+ возвращает `True` , если оба выражения истинны
- 2) возвращает `True` , если хотя бы одно выражение истинно
- 3) возвращает `False` всегда
- 4) присваивает значение

20. Что делает оператор `or`?

- 1)+ возвращает `True` , если хотя бы одно выражение истинно
- 2) возвращает `True` только если оба выражения истинны
- 3) возвращает `False` всегда
- 4) присваивает значение

21. Как проверить, что x не равно 0?

- 1) $x \neq 0$
- 2)+ $x \neq 0$
- 3) $x <> 0$
- 4) $x \neq 0$

22. Что выведет код:

```
python
x = 5
if x < 3:
    print("A")
elif x < 10:
    print("B")
else:
    print("C")
```

- 1) A
- 2)+ B
- 3) C

23. Какой вариант корректен для проверки диапазона чисел?

- 1) `if x>0 and x<10`
- 2)+ `if 0 < x < 10`
- 3) `if x>0 || x<10`
- 4) `if x>0 to 10`

24. Что выведет код:

```
python
x = True
if not x:
    print("A")
else:
    print("B")
```

- 1) A
- 2)+ B

25. Что выведет код:

```
python
x = False
if x or True:
    print("Yes")
else:
    print("No")
```

- 1)+ Yes
- 2) No

26. Какой вариант проверяет, что x меньше 5 или y больше 10?

- 1) `x < 5 & y > 10`
- 2)+ `x < 5 or y > 10`
- 3) `x < 5 && y > 10`
- 4) `x < 5 and y > 10`

27. Что выведет код:

```
python
x = 7
if x % 2 == 0:
    print("even")
elif x % 2 != 0:
    print("odd")
```

- 1) even
- 2)+ odd

28. Что делает блок `elif` в Python?

- 1) выполняется всегда
- 2)+ выполняется, если предыдущие условия ложны
- 3) завершает программу
- 4) присваивает значение

29. Какой оператор проверяет принадлежность элемента коллекции?

- 1) `in`
- 2)+ `in`
- 3) `is`
- 4) `==`

30. Как проверить, что `x` не принадлежит списку `[1, 2, 3]` ?

- 1) `x not [1, 2, 3]`
- 2)+ `x not in [1, 2, 3]`
- 3) `x != [1, 2, 3]`
- 4) `x <> [1, 2, 3]`

31. Что выведет код:

```
python
x = 5
if x > 10:
    print("A")
elif x > 3:
    print("B")
else:
    print("C")
```

- 1) A
- 2)+ B
- 3) C

32. Какой вариант корректно проверяет, что `x` равен 5 и `y` равен 10 одновременно?

- 1) `x = 5 and y = 10`
- 2)+ `x == 5 and y == 10`
- 3) `x = 5 && y = 10`
- 4) `x == 5 & y == 10`

Тема 3. Циклы.

1. Какой цикл выполняется, пока условие истинно?

- 1) `for`
- 2)+ `while`
- 3) `do-while`
- 4) `loop`

2. Что выведет код: `for i in range(3): print(i)`

- 1)+ `0 1 2`
- 2) `1 2 3`
- 3) `0 1 2 3`
- 4) `3 2 1`

3. Какой оператор используется для выхода из цикла?

- 1) `continue`
- 2)+ `break`
- 3) `pass`
- 4) `exit`

4. Что делает оператор `continue` ?

- 1) завершает цикл
- 2)+ пропускает текущую итерацию
- 3) повторяет цикл
- 4) вызывает ошибку

5. Какой диапазон генерирует `range(2, 10, 2)` ?

- 1) `2, 3, 4, 5, 6, 7, 8, 9, 10`
- 2)+ `2, 4, 6, 8`
- 3) `2, 4, 6, 8, 10`
- 4) `2, 3, 4, 5, 6, 7, 8, 9`

6. Как правильно написать бесконечный цикл?

- 1) `for True:`
- 2)+ `while True:`
- 3) `loop True:`
- 4) `repeat True:`

7. Что выведет код:

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

- 1)+ `0 1 2`
- 2) `1 2 3`
- 3) `0 1 2 3`
- 4) `3 2 1`

8. Что делает `for i in "abc": print(i)`?

- 1)+ выводит `a b c`
- 2) выводит `abc`
- 3) выводит `0 1 2`
- 4) вызывает ошибку

9. Что будет результатом:

```
python
for i in range(5):
    if i == 3:
break
    print(i)
```

- 1)+ `0 1 2`
- 2) `0 1 2 3 4`
- 3) `3 4`
- 4) `0 1 2 3`

10. Что будет результатом:

```
python
for i in range(5):
    if i == 3:
        continue
print(i)
```

- 1)+ `0 1 2 4`
- 2) `0 1 2 3 4`
- 3) `3 4`
- 4) `0 1 2 3`

11. Какой из циклов может иметь `else` блок?

- 1) `do-while`
- 2)+ `for` и `while`
- 3) только `for`
- 4) только `while`

12. Что делает блок `else` у цикла?

- 1)+ выполняется, если цикл завершился без `break`
- 2) выполняется всегда
- 3) выполняется только при ошибке
- 4) завершает программу

13. Что выведет код:

python

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

- 1)+ 0 0 0 1 1 0 1 1 2 0 2 1
- 2) 0 0 1 1 2 2
- 3) 0 1 2 3
- 4) 0 0 1 0 2 0

14. Как правильно перебирать элементы списка `lst = [1,2,3]` ?

- 1) `for i in range(lst):`
- 2)+ `for i in lst:`
- 3) `for i = 0; i < lst:`
- 4) `foreach lst as i:`

15. Какой метод списка используется для перебора индексов?

- 1) `items()`
- 2) `values()`
- 3)+ `range(len(lst))`
- 4) `keys()`

16. Что выведет код:

python

```
i = 0  
while i < 3:  
    print(i)  
    i += 2
```

- 1)+ 0 2
- 2) 0 1 2
- 3) 0 1 2 3
- 4) 1 2 3

17. Какой результат:

python

```
for i in range(1,6):  
    if i%2==0:  
        continue  
    print(i)
```

- 1)+ 1 3 5
- 2) 2 4 6
- 3) 1 2 3 4 5
- 4) 0 2 4

18. Что делает `enumerate(lst)` ?

- 1)+ возвращает пары (индекс, значение)
- 2) возвращает только индексы
- 3) возвращает только значения
- 4) вызывает ошибку

19. Как выйти из вложенного цикла сразу?

- 1) `continue`
- 2)+ `break` (во внешнем цикле или через флаг)
- 3) `pass`
- 4) `exit`

20. Что делает `while not done:` ?

- 1) выполняется, если `done = True`
- 2)+ выполняется, если `done = False`
- 3) выполняется всегда
- 4) вызывает ошибку

21. Какой результат:

```
java
for i in range(3):
    print(i)
else:
    print("Done")
```

- 1)+ `0 1 2 Done`
- 2) `0 1 2`
- 3) `Done`
- 4) `1 2 3`

22. Что выведет код:

```
python
for i in range(3):
    if i == 5:
        break
    else:
        print("No break")
```

- 1)+ `No break`
- 2) `0 1 2`
- 3) `5`
- 4) `Error`

23. Какой цикл перебирает элементы словаря?

- 1) `while`
- 2)+ `for key in dict:`
- 3) `do-while`
- 4) `loop`

24. Какой результат:

```
python
lst = [1,2,3]
for i in lst:
    i += 1
    print(lst)
```

- 1)+ [1,2,3]
- 2) [2,3,4]
- 3) [1,2,3,4]
- 4) [0,1,2]

25. Какой способ корректно перебирать индексы и значения списка?

- 1) `for i in lst, v in enumerate(lst):`
- 2)+ `for i,v in enumerate(lst):`
- 3) `for i,v in range(lst):`
- 4) `for i,v in lst.items():`

26. Что делает `break` внутри `while` ?

- 1) пропускает текущую итерацию
- 2)+ завершает цикл
- 3) начинает цикл заново
- 4) вызывает ошибку

27. Какой результат:

```
python
for i in range(3):
    for j in range(2):
        if j == 1:
            break
        print(i,j)
```

- 1)+ 0 0 1 0 2 0
- 2) 0 0 0 1 1 0 1 1
- 3) 0 1 2
- 4) 0 0 1 1 2 2

28. Что выведет код:

```
python
i = 0
while i < 5:
    i += 1
    if i == 3:
        continue
    print(i)
```

- 1)+ 1 2 4 5
- 2) 1 2 3 4 5
- 3) 3 4 5
- 4) 1 2 3 5

29. Какой цикл подходит для перебора символов строки?

- 1) while
- 2)+ for
- 3) loop
- 4) do-while

30. Какой результат:

```
python
for i in range(0):
    print(i)
else:
    print("Empty")
```

- 1)+ Empty
- 2) 0
- 3) Error
- 4) ничего

31. Как корректно завершить вложенный цикл из внутреннего уровня?

- 1) continue
- 2)+ break + флаг
- 3) exit
- 4) pass

32. Как правильно использовать else с for ?

- 1) выполняется всегда
- 2)+ выполняется, если цикл не был прерван break
- 3) выполняется только при ошибке
- 4) выполняется один раз перед циклом

33. Что выведет код:

python

```
for i in range(1,5):  
    if i % 2 == 0:  
        continue  
    print(i)
```

1)+ 1 3

2) 2 4

3) 1 2 3 4

4) 1 2 3

Тема 4. Функции.

1. Как объявить функцию в Python?

- 1) `function my_func():`
- 2)+ `def my_func():`
- 3) `func my_func():`
- 4) `define my_func():`

2. Как вернуть значение из функции?

- 1) `output`
- 2) `print`
- 3)+ `return`
- 4) `yield`

3. Что делает `*args` в функции?

- 1) передаёт ключевые аргументы
- 2)+ передаёт произвольное количество позиционных аргументов
- 3) создаёт список
- 4) создаёт словарь

4. Что делает `**kwargs` в функции?

- 1) передаёт произвольные позиционные аргументы
- 2)+ передаёт произвольные именованные аргументы
- 3) создаёт кортеж
- 4) создаёт множество

5. Как вызвать функцию `foo` ?

- 1) `call foo()`
- 2)+ `foo()`
- 3) `run foo()`
- 4) `exec foo`

6. Что будет выведено:

```
python
def f():
    return 5
print(f())
```

- 1)+ 5
- 2) None
- 3) f
- 4) Error

7. Что вернёт функция без `return` ?

- 1) 0
- 2) ""
- 3)+ None
- 4) False

8. Как объявить функцию с аргументом по умолчанию?

- 1) `def f(x): x=5`
- 2)+ `def f(x=5):`
- 3) `def f(x?:5):`
- 4) `def f(default x=5):`

9. Что делает `lambda x: x*2` ?

- 1) создаёт цикл
- 2) создаёт класс
- 3)+ создаёт анонимную функцию
- 4) создаёт список

10. Как вызвать lambda-функцию `f = lambda x: x+1` для `x=5` ?

- 1) `f.call(5)`
- 2)+ `f(5)`
- 3) `call(f,5)`
- 4) `f.execute(5)`

11. Что выведет:

```
python
def f(a, b=2):
    print(a,b)
f(5)
```

- 1)+ `5 2`
- 2) `5 None`
- 3) `2 5`
- 4) `5 5`

12. Что делает `return` в функции?

- 1) выводит значение на экран
- 2)+ возвращает значение функции
- 3) завершает цикл
- 4) создаёт переменную

13. Как объявить функцию с произвольным количеством аргументов?

- 1) `def f(*):`
- 2)+ `def f(*args, **kwargs):`
- 3) `def f(...):`
- 4) `def f(args, kwargs):`

14. Что выведет:

```
python
def f(x):
    x += 1
    return x
print(f(5))
```

- 1)+ 6
- 2) 5
- 3) None
- 4) 1

15. Как вызвать функцию внутри другой функции?

- 1) нельзя
- 2)+ просто вызвать её по имени
- 3) через `return`
- 4) через `global`

16. Что выведет:

```
python
def f(x):
    return x*2
print(f(3))
```

- 1)+ 6
- 2) 3
- 3) 9
- 4) None

17. Как объявить функцию без аргументов?

- 1) `def f(x=None):`
- 2)+ `def f():`
- 3) `def f(void):`
- 4) `def f(nil):`

18. Что делает `pass` в теле функции?

- 1) возвращает `None`
- 2)+ ничего, используется как заглушка
- 3) завершает функцию
- 4) выводит сообщение

19. Как проверить, что объект является функцией?

- 1) `isinstance(obj, int)`
- 2)+ `callable(obj)`
- 3) `type(obj) == func`
- 4) `hasattr(obj, 'func')`

20. Что вернёт:

```
python
def f():
    return "Hello"
print(f())
```

- 1)+ Hello
- 2) "Hello"
- 3) None
- 4) Error

21. Как объявить рекурсивную функцию?

- 1) нельзя
- 2)+ `def f(): f()`
- 3) `def f(): call(f)`
- 4) `def f(): return f()`

22. Какой результат:

```
python
def f(x): return x+1
print(f(f(2)))
```

- 1)+ 4
- 2) 3
- 3) 2
- 4) 5

23. Как вернуть несколько значений из функции?

- 1) `return x+y`
- 2)+ `return x,y`
- 3) `return [x+y]`
- 4) `return (x+y)`

24. Что делает * при вызове функции `f(*lst)` ?

- 1) создаёт кортеж
- 2)+ распаковывает список как аргументы
- 3) создаёт список
- 4) создаёт словарь

25. Что делает ** при вызове функции `f(**d)` ?

- 1) распаковывает список
- 2)+ распаковывает словарь как именованные аргументы
- 3) создаёт словарь
- 4) создаёт кортеж

26. Какой результат:

python

```
def f(a, b=2):  
    return a+b  
print(f(3))
```

- 1)+ 5
- 2) 3
- 3) 2
- 4) None

27. Какой вариант корректно передаёт функцию как аргумент?

- 1) f(f)
- 2)+ f(f)
- 3) call(f)
- 4) exec(f)

28. Что делает return в рекурсивной функции?

- 1) вызывает ошибку
- 2)+ возвращает значение на каждом уровне рекурсии
- 3) завершает цикл
- 4) печатает значение

29. Как объявить анонимную функцию с двумя аргументами?

- 1) lambda x: y: x+y
- 2)+ lambda x, y: x+y
- 3) def(x, y): x+y
- 4) func(x, y): x+y

30. Как правильно вызвать функцию с аргументами по ключу?

- 1) f(x:5, y:10)
- 2)+ f(x=5, y=10)
- 3) f({x:5, y:10})
- 4) f(5:x, 10:y)

31. Какой результат:

python

```
def f(x):  
    return x*2  
print(f(0))
```

- 1)+ 0
- 2) None
- 3) Error
- 4) 2

32. Как вернуть список из функции?

- 1) `print([1, 2, 3])`
- 2)+ `return [1, 2, 3]`
- 3) `list(1, 2, 3)`
- 4) `return 1, 2, 3`

33. Что делает `yield` в функции?

- 1) возвращает значение и завершает функцию
- 2)+ создаёт генератор
- 3) печатает значение
- 4) создаёт список

Тема 5. Списки, кортежи, словари, множества.

1. Как создать список `lst` с элементами `1, 2, 3` ?

- 1) `lst = (1, 2, 3)`
- 2)+ `lst = [1, 2, 3]`
- 3) `lst = {1, 2, 3}`
- 4) `lst = <1, 2, 3>`

2. Как добавить элемент `4` в список `lst`?

- 1) `lst.add(4)`
- 2)+ `lst.append(4)`
- 3) `lst.insert(4)`
- 4) `lst.push(4)`

3. Как удалить элемент с индексом `1`?

- 1) `lst.remove(1)`
- 2)+ `del lst[1]`
- 3) `lst.pop(2)`
- 4) `lst.delete(1)`

4. Как получить длину списка `lst`?

- 1) `lst.len()`
- 2)+ `len(lst)`
- 3) `lst.length`
- 4) `lst.size()`

5. Как создать кортеж `t` с элементами `1, 2, 3`?

- 1) `t = [1, 2, 3]`
- 2) `t = {1, 2, 3}`
- 3)+ `t = (1, 2, 3)`
- 4) `t = <1, 2, 3>`

6. Кортеж отличается от списка тем, что:

- 1) его нельзя печатать
- 2) он всегда пустой
- 3)+ он неизменяемый
- 4) он хранит только числа

7. Как создать словарь с ключами `'a'`, `'b'` и значениями `1, 2`?

- 1) `d = [("a", 1), ("b", 2)]`
- 2)+ `d = {'a': 1, 'b': 2}`
- 3) `d = ('a': 1, 'b': 2)`
- 4) `d = a:1, b:2`

8. Как получить значение по ключу `'a'`?

- 1) `d.get('b')`
- 2)+ `d['a']`
- 3) `d.a`
- 4) `d('a')`

9. Как проверить, есть ли ключ 'c' в словаре d?

- 1) `'c' in d.values()`
- 2)+ `'c' in d`
- 3) `'c' in d.items()`
- 4) `d.has('c')`

10. Как удалить ключ 'a' из словаря?

- 1) `d.popitem('a')`
- 2)+ `d.pop('a')`
- 3) `del d.values('a')`
- 4) `d.remove('a')`

11. Как создать множество с элементами 1, 2, 3 ?

- 1) `s = [1, 2, 3]`
- 2) `s = (1, 2, 3)`
- 3)+ `s = {1, 2, 3}`
- 4) `s = set[]`

12. Как добавить элемент 4 в множество?

- 1) `s.push(4)`
- 2)+ `s.add(4)`
- 3) `s.append(4)`
- 4) `s.insert(4)`

13. Как удалить элемент 2 из множества?

- 1) `s.remove_at(2)`
- 2)+ `s.remove(2)`
- 3) `del s[2]`
- 4) `s.pop(2)`

14. Что делает `s1 & s2` для множеств?

- 1) объединяет множества
- 2)+ возвращает пересечение
- 3) возвращает разность
- 4) проверяет равенство

15. Что делает `s1 | s2` для множеств?

- 1)+ объединяет множества
- 2) возвращает пересечение
- 3) возвращает разность
- 4) проверяет равенство

16. Что делает `s1 - s2` ?

- 1) пересечение
- 2) объединение
- 3)+ разность множеств
- 4) симметричная разность

17. Как преобразовать список в кортеж?

- 1) `tuple(list)`
- 2)+ `tuple(lst)`
- 3) `list(tuple)`
- 4) `tuple[list]`

18. Как преобразовать кортеж в список?

- 1) `list(tuple)`
- 2)+ `list(t)`
- 3) `tuple(list)`
- 4) `t.list()`

19. Как получить индекс элемента `2` в списке `lst = [1, 2, 3]` ?

- 1) `lst.find(2)`
- 2)+ `lst.index(2)`
- 3) `lst.loc(2)`
- 4) `lst.position(2)`

20. Как проверить, есть ли элемент `3` в списке?

- 1) `lst.contains(3)`
- 2)+ `3 in lst`
- 3) `lst.has(3)`
- 4) `lst.check(3)`

21. Как отсортировать список `lst = [3, 1, 2]` ?

- 1) `sorted(lst)`
- 2)+ `lst.sort()`
- 3) `lst.order()`
- 4) `lst.sorted()`

22. Как создать копию списка?

- 1) `lst.copy_list()`
- 2)+ `lst.copy()`
- 3) `copy(lst)`
- 4) `lst[:]`

23. Как удалить последний элемент списка?

- 1) `lst.remove(-1)`
- 2)+ `lst.pop()`
- 3) `del lst[-1]`
- 4) `lst.delete()`

24. Что делает `d.keys()` ?

- 1)+ возвращает ключи словаря
- 2) возвращает значения
- 3) возвращает пары ключ-значение
- 4) удаляет ключи

25. Что делает `d.values()` ?

- 1)+ возвращает значения словаря
- 2) возвращает ключи
- 3) возвращает пары
- 4) удаляет значения

26. Что делает `d.items()`?

- 1)+ возвращает пары ключ-значение
- 2) возвращает только ключи
- 3) возвращает только значения
- 4) удаляет пары

27. Как объединить два списка `a` и `b`?

- 1) `a & b`
- 2) `a * b`
- 3)+ `a + b`
- 4) `a | b`

28. Как удалить все элементы множества `s`?

- 1) `s.clear_all()`
- 2)+ `s.clear()`
- 3) `del s[:]`
- 4) `s.remove_all()`

29. Как проверить, является ли множество пустым?

- 1) `s.empty()`
- 2)+ `len(s) == 0`
- 3) `s.is_empty()`
- 4) `s == None`

30. Как создать множество из списка `lst` ?

- 1) `set(lst)`
- 2)+ `set(lst)`
- 3) `lst.set()`
- 4) `list(set)`

Тема 6. Строки.

1. Как получить длину строки `s`?

- 1) `s.length()`
- 2)+ `len(s)`
- 3) `s.len()`
- 4) `length(s)`

2. Как получить первый символ строки `s = "Python"` ?

- 1) `s[0:1]`
- 2)+ `s[0]`
- 3) `s.first()`
- 4) `s[:0]`

3. Как получить последние два символа строки?

- 1) `s[-2:]`
- 2)+ `s[-2:]`
- 3) `s[2:]`
- 4) `s[:-2]`

4. Как объединить строки `a = "Hello"` и `b = "World"` ?

- 1) `a & b`
- 2)+ `a + b`
- 3) `a.concat(b)`
- 4) `a.join(b)`

5. Как разделить строку по пробелам?

- 1) `s.split(",")`
- 2)+ `s.split()`
- 3) `s.split(" ")`
- 4) `s.separate()`

6. Как объединить список строк `lst` в одну строку через пробел?

- 1) `" ".join(lst)`
- 2)+ `" ".join(lst)`
- 3) `lst.join(" ")`
- 4) `join(lst, " ")`

7. Как заменить подстроку "a" на "b"?

- 1) `s.change("a", "b")`
- 2)+ `s.replace("a", "b")`
- 3) `s.update("a", "b")`
- 4) `s.sub("a", "b")`

8. Как проверить, состоит ли строка только из цифр?

- 1) `s.isdigit()`
- 2)+ `s.isdigit()`
- 3) `s.isnumeric()`
- 4) `s.onlydigits()`

9. Как проверить, состоит ли строка только из букв?

- 1) `s.isalpha()`
- 2)+ `s.isalpha()`
- 3) `s.isletter()`
- 4) `s.alpha()`

10. Как удалить пробелы в начале и конце строки?

- 1) `s.strip(" ")`
- 2)+ `s.strip()`
- 3) `s.trim()`
- 4) `s.clean()`

11. Как перевести строку в верхний регистр?

- 1) `s.uppercase()`
- 2)+ `s.upper()`
- 3) `s.toUpper()`
- 4) `s.cap()`

12. Как перевести строку в нижний регистр?

- 1) `s.lowercase()`
- 2)+ `s.lower()`
- 3) `s.toLower()`
- 4) `s.small()`

13. Что делает `s.startswith("Py")` ?

- 1)+ проверяет, начинается ли строка с "Py"
- 2) проверяет, оканчивается ли строка на "Py"
- 3) ищет "Py" в строке
- 4) заменяет "Py"

14. Что делает `s.endswith("on")` ?

- 1)+ проверяет, оканчивается ли строка на "on"
- 2) проверяет, начинается ли строка с "on"
- 3) ищет "on" в строке
- 4) заменяет "on"

15. Как получить индекс первой буквы "t" в "Python"?

- 1) `s.find(1, "t")`
- 2)+ `s.find("t")`
- 3) `s.index(1, "t")`
- 4) `s.loc("t")`

16. Что делает `s.count("o")` ?

- 1) возвращает позицию
- 2)+ считает количество "o"
- 3) заменяет "o"
- 4) удаляет "o"

17. Как проверить, состоит ли строка только из пробелов?

- 1) `s.isspace()`
- 2)+ `s.isspace()`
- 3) `s.space()`
- 4) `s.isblank()`

18. Что делает `s.splitlines()` ?

- 1) разделяет по пробелам
- 2)+ разделяет по строкам
- 3) объединяет строки
- 4) удаляет переносы

19. Как правильно использовать f-строку для переменной name?

- 1) `f("Hello {name}")`
- 2)+ `f"Hello {name}"`
- 3) `"Hello {name}"`
- 4) `"Hello "+name+""`

20. Как обрезать строку с позиции 2 до 5?

- 1) `s.cut(2,5)`
- 2)+ `s[2:5]`
- 3) `s.slice(2,5)`
- 4) `s[2..5]`

21. Как проверить, начинается ли строка с заглавной буквы?

- 1) `s.isupper()`
- 2)+ `s.istitle()`
- 3) `s.iscapital()`
- 4) `s.isupperfirst()`

22. Как проверить, состоит ли строка из букв и цифр?

- 1) `s.isalnum()`
- 2)+ `s.isalnum()`
- 3) `s.isalpha()`
- 4) `s.isdigit()`

23. Как объединить строки списка `lst` без разделителя?

- 1) `lst.join("")`
- 2)+ `"".join(lst)`
- 3) `join("", lst)`
- 4) `"".concat(lst)`

24. Как получить подстроку от индекса 3 до конца?

- 1) `s[0:3]`
- 2)+ `s[3:]`
- 3) `s[3:len(s)-1]`
- 4) `s[3..]`

25. Как перевернуть строку s?

- 1) `s.reverse()`
- 2)+ `s[::-1]`
- 3) `s.flip()`
- 4) `reversed(s)`

26. Как проверить, состоит ли строка только из цифр с пробелами?

- 1) `s.isdigit()`
- 2)+ `s.replace(" ", "").isdigit()`
- 3) `s.isnumeric()`
- 4) `s.isdigit(True)`

27. Как заменить все пробелы на подчеркивания?

- 1) `s.replace(" ", "_")`
- 2)+ `s.replace(" ", "_")`
- 3) `s.sub(" ", "_")`
- 4) `s.change(" ", "_")`

28. Как преобразовать строку в список символов?

- 1) `list(s)`
- 2)+ `list(s)`
- 3) `s.split("")`
- 4) `s.toList()`

29. Как удалить все пробелы в начале строки s?

- 1) `s.strip()`
- 2)+ `s.lstrip()`
- 3) `s.rstrip()`
- 4) `s.trimLeft()`

30. Как удалить все пробелы в конце строки s?

- 1) `s.strip()`
- 2) `s.lstrip()`
- 3)+ `s.rstrip()`
- 4) `s.trimRight()`

31. Как проверить, содержит ли строка подстроку "Py"?

- 1) `s.startswith("Py")`
- 2) `s.endswith("Py")`
- 3)+ `"Py" in s`
- 4) `s.has("Py")`

32. Как сделать первую букву строки заглавной?

- 1) `s.capitalize()`
- 2)+ `s.capitalize()`
- 3) `s.upperfirst()`
- 4) `s.titlecase()`

33. Как преобразовать строку "hello world" в "Hello World" ?

- 1) `s.capitalize()`
- 2)+ `s.title()`
- 3) `s.upper()`
- 4) `s.capitalize_all()`

Тема 7. Файлы, модули.

1. Как открыть файл data.txt для чтения?

- 1) `open("data.txt", "w")`
- 2)+ `open("data.txt", "r")`
- 3) `open("data.txt", "rw")`
- 4) `open("data.txt", "a")`

2. Как открыть файл для записи с перезаписью?

- 1) `"r"`
- 2)+ `"w"`
- 3) `"a"`
- 4) `"rw"`

3. Как открыть файл для добавления данных?

- 1) `"r"`
- 2) `"w"`
- 3)+ `"a"`
- 4) `"rw"`

4. Как закрыть файл f?

- 1) `f.exit()`
- 2)+ `f.close()`
- 3) `close(f)`
- 4) `f.end()`

5. Как прочитать весь файл в строку?

- 1) `f.readline()`
- 2)+ `f.read()`
- 3) `f.readlines()`
- 4) `read(f)`

6. Как прочитать файл построчно в список?

- 1) `f.read()`
- 2)+ `f.readlines()`
- 3) `f.readline()`
- 4) `f.readall()`

7. Как прочитать одну строку из файла?

- 1) `f.read()`
- 2) `f.readlines()`
- 3)+ `f.readline()`
- 4) `f.next()`

8. Как записать строку "Hello" в файл f?

- 1) `f.write_line("Hello")`
- 2)+ `f.write("Hello")`
- 3) `f.writeline("Hello")`
- 4) `f.add("Hello")`

9. Как использовать менеджер контекста для файла?

- 1) `file.open("f.txt") as f:`
- 2)+ `with open("f.txt") as f:`
- 3) `try open("f.txt") as f:`
- 4) `open("f.txt") as f:`

10. Как проверить, существует ли модуль `math` перед импортом?

- 1) `hasmodule("math")`
- 2)+ `importlib.util.find_spec("math")`
- 3) `import math?`
- 4) `exists(math)`

11. Как импортировать все функции модуля `math`?

- 1) `import math.*`
- 2)+ `from math import *`
- 3) `include math`
- 4) `import all from math`

12. Как импортировать только `sqrt` из `math` ?

- 1) `import math.sqrt`
- 2)+ `from math import sqrt`
- 3) `import sqrt from math`
- 4) `include math.sqrt`

13. Как получить список функций и переменных модуля `math` ?

- 1) `dir(math)`
- 2)+ `dir(math)`
- 3) `list(math)`
- 4) `vars(math)`

14. Как получить путь модуля `math` ?

- 1) `math.path`
- 2) `math.file`
- 3)+ `math.file` (если не встроенный)
- 4) `getpath(math)`

15. Как проверить, встроенный ли модуль `sys` ?

- 1) `sys.isbuiltin()`
- 2)+ `sys.builtin_module_names`
- 3) `builtin(sys)`
- 4) `sys.exists()`

16. Что делает `import os` ?

- 1) открывает файл
- 2) создаёт объект
- 3)+ импортирует модуль `os`
- 4) выполняет скрипт `os`

17. Как выполнить скрипт script.py из другого файла?

- 1) `run("script.py")`
- 2) `exec("script.py")`
- 3)+ `import script`
- 4) `include "script.py"`

18. Как получить текущую директорию?

- 1) `os.path`
- 2) `os.dir()`
- 3)+ `os.getcwd()`
- 4) `os.current()`

19. Как создать новую директорию test?

- 1) `os.mkdir("test", overwrite=True)`
- 2)+ `os.mkdir("test")`
- 3) `os.make("test")`
- 4) `os.create("test")`

20. Как удалить файл data.txt?

- 1) `os.remove_file("data.txt")`
- 2)+ `os.remove("data.txt")`
- 3) `os.delete("data.txt")`
- 4) `os.erase("data.txt")`

21. Как переименовать файл old.txt в new.txt?

- 1) `os.rename_file("old.txt", "new.txt")`
- 2)+ `os.rename("old.txt", "new.txt")`
- 3) `os.move("old.txt", "new.txt")`
- 4) `os.change("old.txt", "new.txt")`

22. Как получить список файлов в текущей директории?

- 1) `os.files()`
- 2)+ `os.listdir()`
- 3) `os.dir()`
- 4) `os.getfiles()`

23. Как проверить, существует ли файл data.txt?

- 1) `os.exist("data.txt")`
- 2)+ `os.path.exists("data.txt")`
- 3) `exists("data.txt")`
- 4) `os.check("data.txt")`

24. Как проверить, является ли путь директорией?

- 1) `os.isdir("path")`
- 2)+ `os.path.isdir("path")`
- 3) `os.isdir("path")`
- 4) `path.isdir()`

25. Как проверить, является ли путь файлом?

- 1) `os.isfile("path")`
- 2)+ `os.path.isfile("path")`
- 3) `path.isfile()`
- 4) `os.isfile("path")`

26. Как объединить пути dir и file?

- 1) `dir + "/" + file`
- 2)+ `os.path.join(dir, file)`
- 3) `join(dir, file)`
- 4) `path.concat(dir, file)`

27. Как импортировать модуль с псевдонимом?

- 1) `import math as m`
- 2)+ `import math as m`
- 3) `include math as m`
- 4) `from math import as m`

28. Как узнать версию Python через модуль sys?

- 1) `sys.version()`
- 2)+ `sys.version`
- 3) `sys.getversion()`
- 4) `sys.python_version()`

29. Как проверить наличие переменной в модуле?

- 1) `hasattr(var, "module")`
- 2)+ `hasattr(module, "var")`
- 3) `var in module`
- 4) `module.has("var")`

30. Как импортировать модуль динамически по имени?

- 1) `import "module"`
- 2) `exec("import module")`
- 3)+ `importlib.import_module("module")`
- 4) `include(module)`

31. Как вызвать функцию модуля через псевдоним m?

- 1) `m->func()`
- 2)+ `m.func()`
- 3) `m:func()`
- 4) `call(m.func)`

Тема 8. Исключения, обработка ошибок.

1. Как обработать исключение в Python?

- 1) `try {} catch {}`
- 2)+ `try: ... except:`
- 3) `begin ... rescue`
- 4) `handle ... end`

2. Как поймать только `ZeroDivisionError` ?

- 1) `except Exception:`
- 2)+ `except ZeroDivisionError:`
- 3) `except all:`
- 4) `except ZeroDiv:`

3. Как получить текст ошибки?

- 1) `e.text`
- 2)+ `str(e)`
- 3) `e.message`
- 4) `e.error()`

4. Как поймать несколько исключений?

- 1)+ `except (ValueError, ZeroDivisionError):`
- 2) `except ValueError|ZeroDivisionError:`
- 3) `except ValueError, ZeroDivisionError:`

5. Что делает `finally` блок?

- 1) выполняется только при ошибке
- 2)+ выполняется всегда
- 3) завершает программу
- 4) создаёт исключение

6. Как сгенерировать исключение вручную?

- 1) `raise Exception()`
- 2)+ `raise Exception()`
- 3) `throw Exception()`
- 4) `error Exception()`

7. Как поймать любую ошибку?

- 1) `except Error:`
- 2)+ `except Exception:`
- 3) `except all:`
- 4) `except:`

8. Как вывести traceback ошибки?

- 1) `print(traceback)`
- 2)+ `import traceback; traceback.print_exc()`
- 3) `traceback.show()`
- 4) `print_exc()`

9. Как определить пользовательское исключение?

- 1) `class MyError: Exception`
- 2)+ `class MyError(Exception): pass`
- 3) `exception MyError`
- 4) `MyError = Exception`

10. Что делает `assert x>0` ?

- 1) проверяет и исправляет x
- 2)+ вызывает `AssertionError` , если `x≤0`
- 3) ничего
- 4) выводит сообщение

11. Как обработать несколько блоков `except` ?

- 1) `except: except:`
- 2)+ `try: ... except ValueError: ... except TypeError:`
- 3) `try: ... except(ValueError, TypeError):`
- 4) `try: ... except all:`

12. Как завершить программу с ошибкой?

- 1) `exit(0)`
- 2)+ `sys.exit("Error")`
- 3) `quit()`
- 4) `stop()`

13. Что делает `raise` без аргументов в `except` ?

- 1) создаёт новый exception
- 2)+ повторно вызывает текущее исключение
- 3) завершает программу
- 4) игнорирует исключение

14. Как поймать ошибки при делении на ноль?

- 1) `except Exception:`
- 2)+ `except ZeroDivisionError:`
- 3) `except DivideError:`
- 4) `except ArithmeticError:`

15. Как выполнить код, даже если ошибка произошла?

- 1) `except`
- 2) `try`
- 3)+ `finally`
- 4) `raise`

16. Как проверить тип исключения?

- 1) `e.type`
- 2)+ `type(e)`
- 3) `e.class`
- 4) `e.kind`

17. Как получить имя исключения?

- 1) `e.name`
- 2)+ `e.class.name`
- 3) `e.type`
- 4) `e.id`

18. Что делает блок `else` в обработке ошибок?

- 1) выполняется всегда
- 2)+ выполняется, если исключения не произошло
- 3) выполняется только при ошибке
- 4) завершает программу

19. Как поймать все ошибки, кроме `KeyboardInterrupt` ?

- 1) `except Exception:`
- 2) `except KeyboardInterrupt:`
- 3)+ `except Exception as e: ... (KeyboardInterrupt отдельно)`
- 4) `except all:`

20. Как выбросить исключение с сообщением "Error"?

- 1) `throw "Error"`
- 2)+ `raise Exception("Error")`
- 3) `error("Error")`
- 4) `fail("Error")`

21. Что делает `try` блок?

- 1) завершает программу
- 2)+ выполняет код, который может вызвать исключение
- 3) создаёт исключение
- 4) игнорирует ошибки

22. Как использовать несколько `except` для одного `try` ?

- 1) `except ValueError & TypeError:`
- 2)+ `except ValueError: ... except TypeError:`
- 3) `except(ValueError, TypeError):`
- 4) `except ValueError|TypeError:`

23. Что выведет код:

```
python
try:
    x = 1/0
except ZeroDivisionError:
    print("Error")
```

- 1)+ `Error`
- 2) `0`
- 3) `Exception`
- 4) `1`

24. Как поймать конкретное сообщение ошибки?

- 1) `str(e)`
- 2)+ `str(e)`
- 3) `e.msg`
- 4) `e.text`

25. Как завершить программу при возникновении ошибки?

- 1) `raise`
- 2)+ `sys.exit("Error")`
- 3) `stop()`
- 4) `end()`

26. Как отловить ошибки при работе с файлами?

- 1) `except FileNotFoundError:`
- 2)+ `except IOError:`
- 3) `except FileNotFoundError:`
- 4) `except Error:`

27. Как безопасно открыть файл и обработать ошибки?

- 1) `f = open("file")`
- 2)+ `try: f = open("file") ... except IOError: ... finally: f.close()`
- 3) `open("file") except`
- 4) `with open("file") except IOError:`

28. Как проверить, поймана ли ошибка?

- 1) `if error:`
- 2)+ `except ... as e:`
- 3) `try ... catch e`
- 4) `check(error)`

29. Как записать traceback в файл?

- 1) `traceback.save("file")`
- 2)+ `import traceback; traceback.print_exc(file=f)`
- 3) `print(traceback, file=f)`
- 4) `log(traceback)`

30. Как пробросить исключение в другой блок?

- 1) `throw e`
- 2)+ `raise`
- 3) `raise e`
- 4) `send(e)`

31. Как обработать деление на ноль и другие ошибки отдельно?

- 1) `except Exception:`
- 2) `except all:`
- 3)+ `except ZeroDivisionError: ... except Exception:`
- 4) `except ArithmeticError:`

32. Как создать пользовательское исключение с сообщением?

- 1) `class MyError: msg`
- 2)+ `class MyError(Exception): pass; raise MyError("message")`
- 3) `throw MyError("message")`
- 4) `raise Error("message")`

33. Что делает `except` без указания ошибки?

- 1) ловит только `Exception`
- 2)+ ловит все исключения
- 3) не ловит ничего
- 4) вызывает ошибку

34. Как поймать несколько исключений и сохранить объект ошибки?

- 1) `except (ValueError, TypeError): e`
- 2)+ `except (ValueError, TypeError) as e`
- 3) `except ValueError, TypeError as e`
- 4) `except ValueError|TypeError as e`

35. Как использовать `try/except` с ресурсами?

- 1) `try: f=open(...) finally:`
- 2)+ `with open(...) as f: try: ... except`
- 3) `open(...) try`
- 4) `f=open(...) except`

generated at geetest.ru

Table of Contents

Тема 1. Основы Python.	2
Тема 2. Условные конструкции.	7
Тема 3. Циклы.	12
Тема 4. Функции.	19
Тема 5. Списки, кортежи, словари, множества.	25
Тема 6. Строки.	29
Тема 7. Файлы, модули.	34
Тема 8. Исключения, обработка ошибок.	38