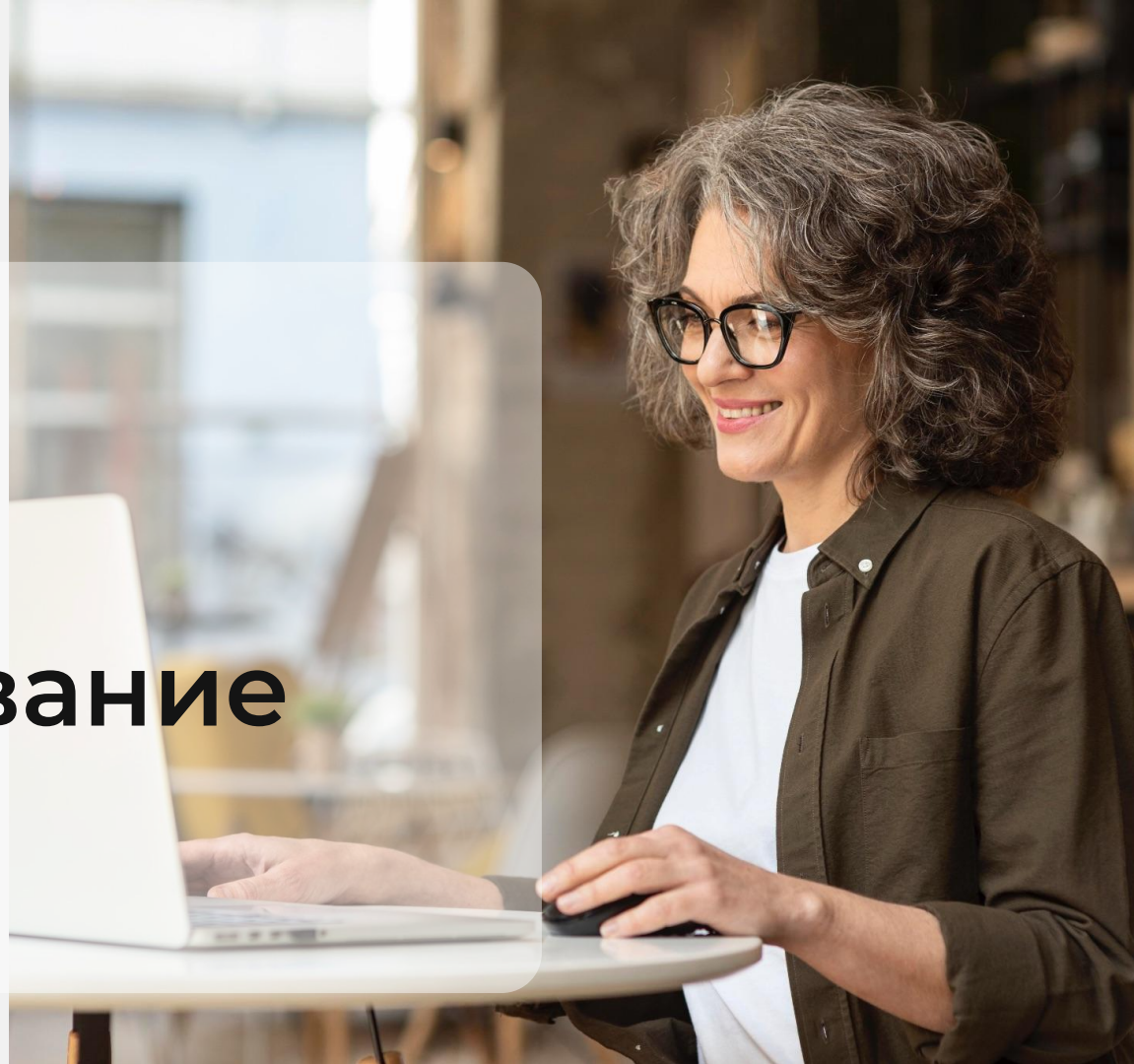
IT CAREER HUB

Python

Строки: форматирование



Преподаватель

Портрет

Имя Фамилия

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы

Самые яркие проекты

Дополнительная информация по вашему усмотрению

Корпоративный e-mail

Социальные сети (по желанию)

Важно

- Камера должна быть включена на протяжении всего занятия
- В течение занятия вопросы задавать в чате или когда преподаватель спрашивает, есть ли у Вас вопросы
- Вести себя уважительно и этично по отношению к остальным участникам занятия
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя

Повторение

- Методы строк
- Методы проверки предиката
- Поиск подстроки в строке
- Методы изменения регистра
- Метод replace
- Методы split и join
- Методы strip, lstrip, rstrip

План занятия

- Форматирование строк
- Задания для закрепления
- Метод format
- Задания для закрепления
- Форматирование чисел
- Задания для закрепления

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

ICH IT CAREER HUB

Форматирование строк

Полезно знать

ICH
IT CAREER HUB

Форматирование строк

Это способ вставки значений переменных или результатов выражений внутрь строки.

C-style форматирование

C-style форматирование строк в Python использует оператор % для вставки значений в строку. Этот способ форматирования напоминает стиль форматирования строк в языке программирования **C** и был одним из первых способов форматирования в Python.

C-style форматирование

Синтаксис

```
"форматирующая строка" % (значения)
```

Пояснения

- форматирующая строка содержит спецификаторы, начинающиеся с символа %
- значения — это переменные или выражения

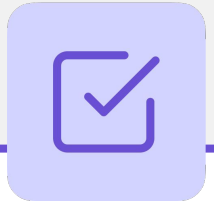
Основные спецификаторы

Метод	Описание
%s	Строка
%d	Целое число
%f	Число с плавающей точкой
%.2f	Число с плавающей точкой, округлённое до 2 знаков после запятой

Примеры

```
# Форматирование строки и целого числа
name = "Alice"
age = 30
text = "My name is %s and I am %d years old." % (name, age)
print(text)

# Форматирование числа с плавающей точкой
pi = 3.14159
text = "The value of pi is approximately %.2f." % pi
print(text)
```

Преобразование типов в C-style



В C-style форматировании можно легко преобразовывать **числа в строки** с помощью спецификаторов формата. Например, использование `%s` автоматически преобразует любое число в строку. Но при этом невозможно преобразовать строку в число, это вызовет ошибку `TypeError`.

Пример

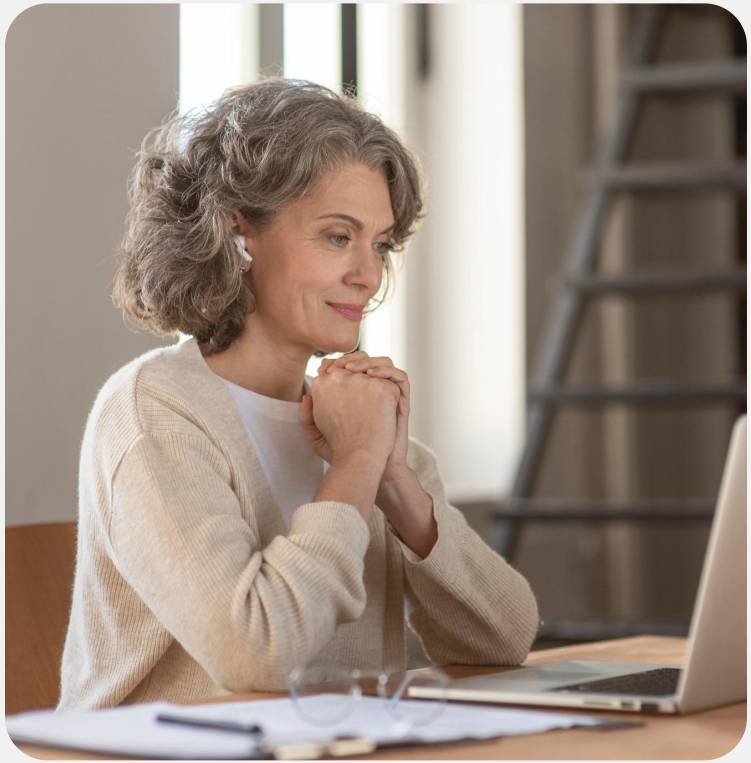
```
num = 42
text = "The number is %s." % num
print(text)

text = "42"  # Число в формате str
formatted = "This will cause an error: %d" % text
print(formatted)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

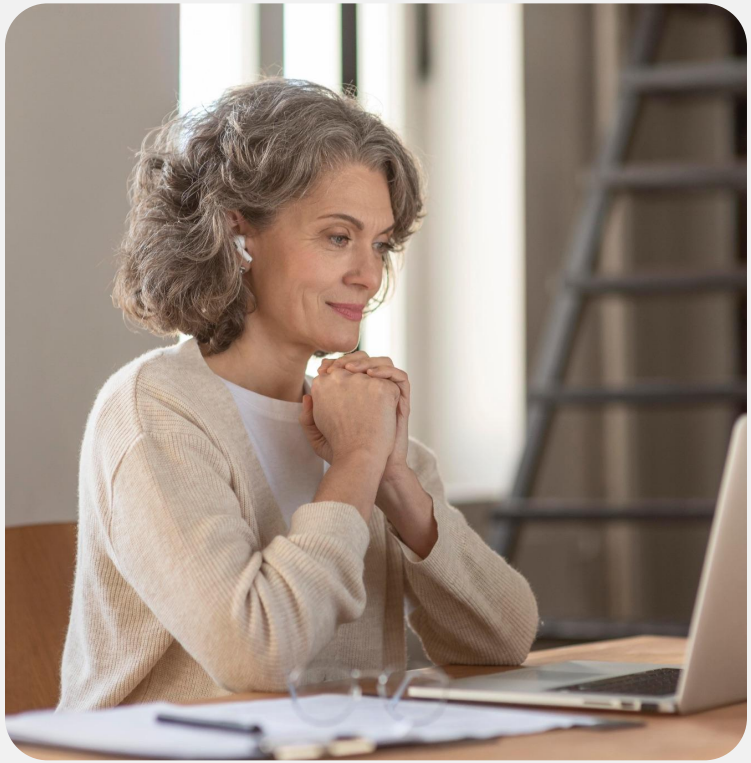


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
name = "Alice"

age = 30

text = "My name is %d and I am %s years old." % (name, age)

print(text)
```

a. "My name is Alice and I am 30 years old."
b. "My name is %s and I am %d years old."
c. Ошибка
d. "My name is Alice and I am %d years old."

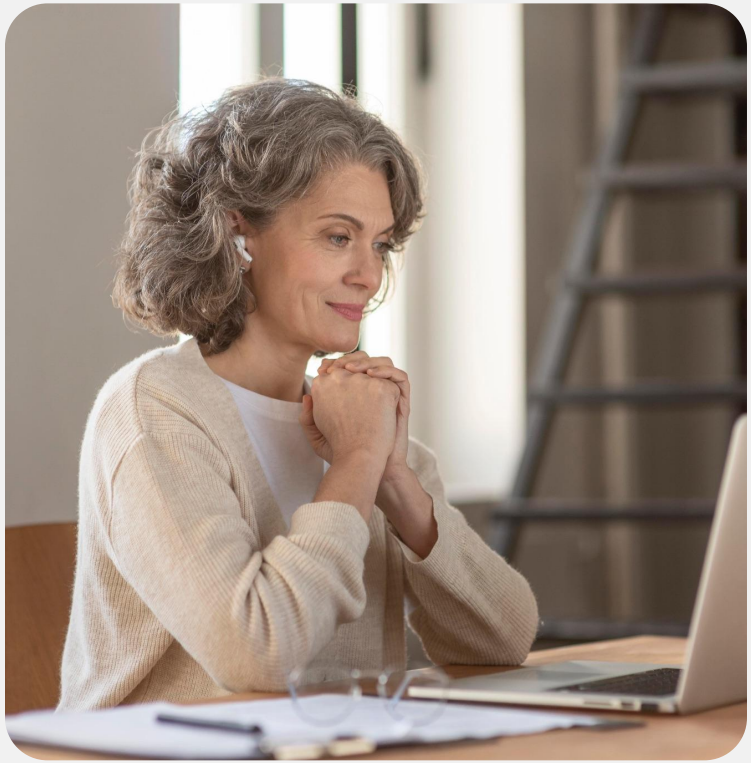


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
name = "Alice"

age = 30

text = "My name is %d and I am %s years old." % (name, age)

print(text)
```

a. "My name is Alice and I am 30 years old."
b. "My name is %s and I am %d years old."
c. Ошибка
d. "My name is Alice and I am %d years old."

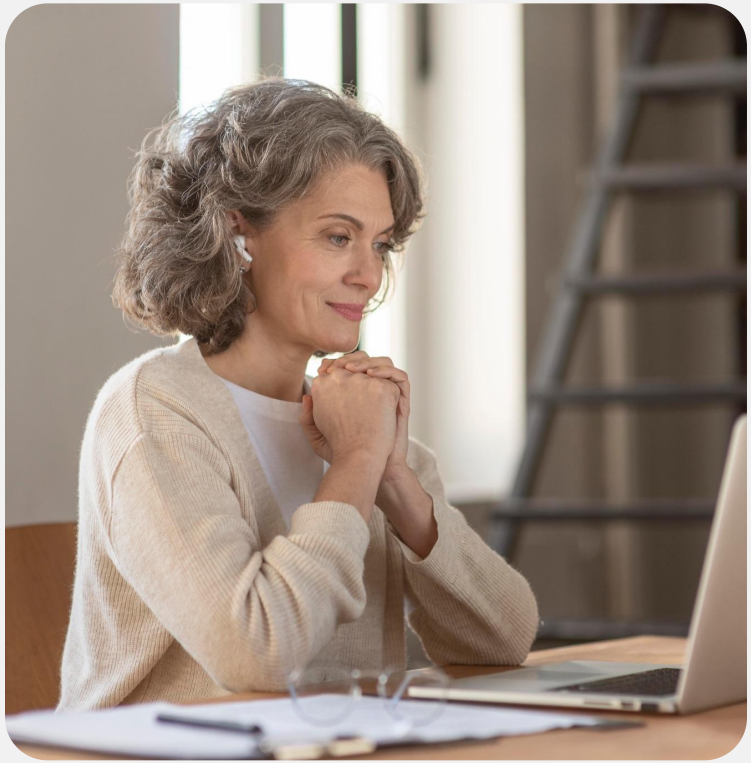


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
pi = 3.14159

text = "The value of pi is approximately
%.2f." % pi

print(text)
```

a. "The value of pi is approximately 3.14."
b. "The value of pi is approximately 3.14159."
c. Ошибка
d. "The value of pi is approximately %.2f."



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
pi = 3.14159

text = "The value of pi is approximately
%.2f." % pi

print(text)
```

a. **"The value of pi is approximately 3.14."**
b. "The value of pi is approximately 3.14159."
c. Ошибка
d. "The value of pi is approximately %.2f."

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Метод format

Полезно знать

ICH
IT CAREER HUB

Метод format

Это гибкий способ форматирования строк, который позволяет вставлять значения в строку с использованием фигурных скобок `{}` в качестве плейсхолдеров.

Метод format()

Синтаксис

```
"строка с {} внутри".format(значение1, значение2, ...)
```

Пояснения

- Внутри строки используются фигурные скобки `{}`, которые будут заменены переданными значениями из метода `format()`
- Можно использовать позиционные аргументы, именованные аргументы.

Позиционные аргументы

```
name = "Alice"

age = 30

text = "My name is {} and I am {} years old."

print(text.format(name, age))
```

Именованные аргументы

Каждому плейсхолдеру можно присвоить имя, что делает форматирование более понятным. При этом можно несколько раз использовать то же имя.

```
text = "My name is {name} and I am {age} years old. Are you also {age} years old?"

print(text.format(name="Bob", age=25))
```

Именованные аргументы

Можно явно указать, какой аргумент вставить в каждое место. Индексы также можно использовать несколько раз.

```
text = "Her name is {0} and she is {1} years old. {0} loves Python."

print(text.format("Anna", 28))
```

Комбинирование позиционных и именованных аргументов

Можно использовать как позиционные, так и именованные аргументы одновременно, что даёт большую гибкость в форматировании строк.

```
text = "The {0} is {color}."

print(text.format("sky", color="blue"))
```

Полезно знать

ICH
IT CAREER HUB

f-строки

Это удобный способ форматирования строк,
который был введён в Python 3.6.

f-строки

Синтаксис

```
f"текст {переменная} текст {выражение}"
```

Пояснения

Чтобы использовать **f-строки**, перед строкой нужно добавить префикс `f`, а переменные или выражения помещаются напрямую в фигурные скобки `{}` внутри строки.

Вставка переменных

```
name = "Alice"

age = 25

text = f"My name is {name} and I am {age} years old."

print(text)
```

Вставка выражений

```
x = 10

y = 20

text = f"The sum of {x} and {y} is {x + y}."

print(text)
```

Вставка вызова функций и методов

```
text = "Python"

text_info = f"The length of '{text}' is {len(text)} and its uppercase version is {text.upper()}."

print(text_info)
```

Вставка в многострочную строку

```
name = "Charlie"

age = 30

text = f"""Info

Name: {name}

Age: {age}

"""

print(text)
```

Преимущества f-строк

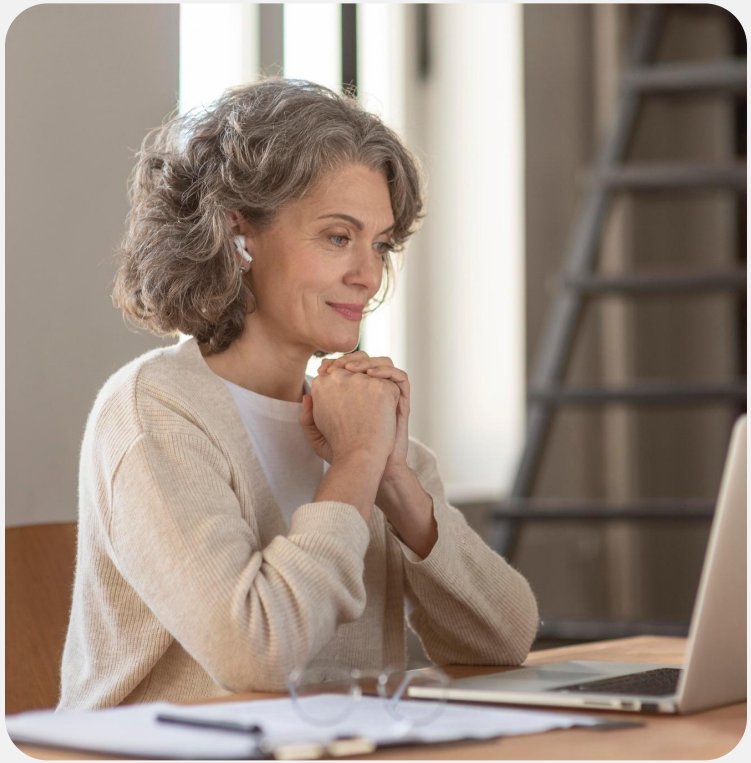
- **Удобство:** Позволяют вставлять переменные и выражения прямо в строку.
- **Читаемость:** Делают код более понятным и лаконичным.
- **Гибкость:** Поддерживают любые выражения внутри `{}`.
- **Быстрота:** Работают быстрее, чем другие способы форматирования строк, такие как `%` и `format()`.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

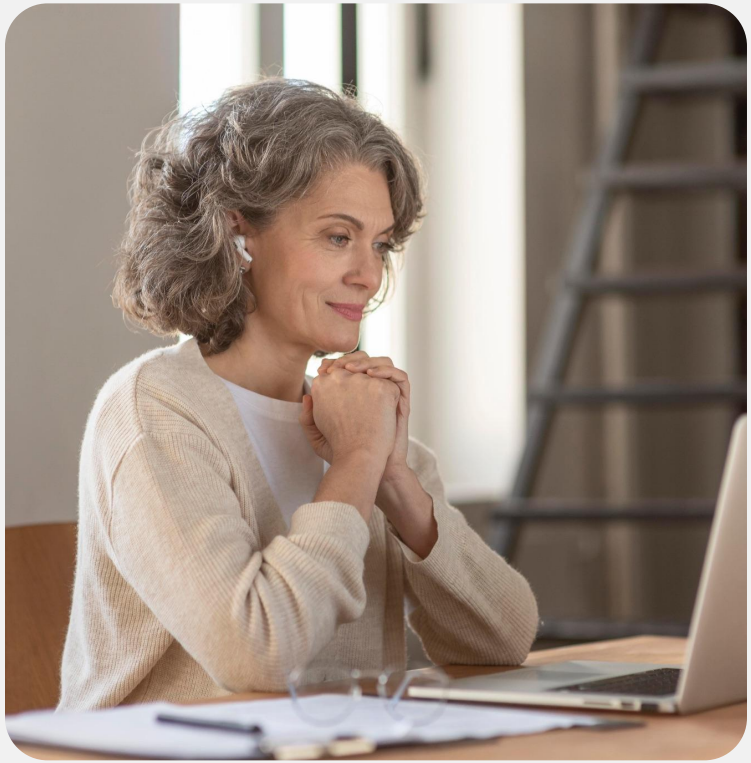


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = f"The sum of {10} and {20} is {10
+ 20}."

print(text)
```

a. "The sum of 10 and 20 is 30."
b. "The sum of 10 and 20 is {10 + 20}."
c. Ошибка
d. "The sum of {10} and {20} is {10 + 20}."

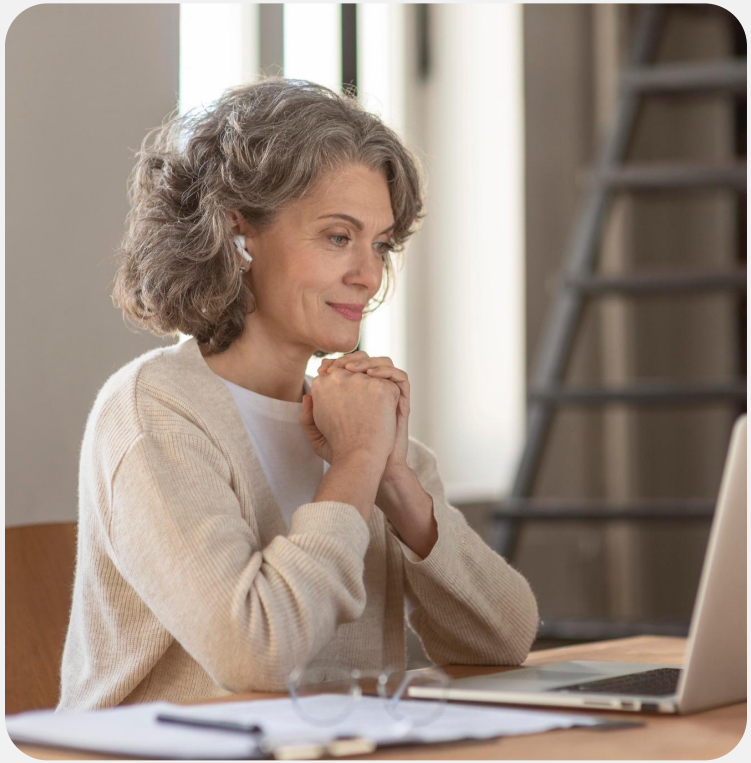


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = f"The sum of {10} and {20} is {10 + 20}."

print(text)
```

a. "The sum of 10 and 20 is 30."
b. "The sum of 10 and 20 is {10 + 20}."
c. Ошибка
d. "The sum of {10} and {20} is {10 + 20}."

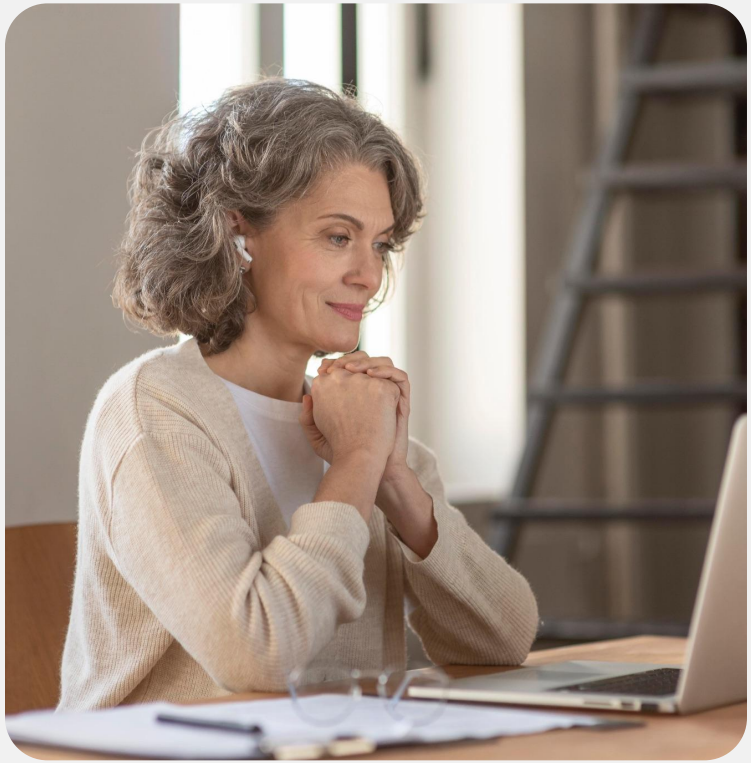


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = "The {0} is
{color}.".format("sky", color="blue")

print(text)
```

a. "The sky is color."
b. "The {0} is {color}."
c. "The sky is blue."
d. "The 0 is color."



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = "The {0} is
{color}.".format("sky", color="blue")

print(text)
```

a. "The sky is color."
b. "The {0} is {color}."
c. "The sky is blue."
d. "The 0 is color."

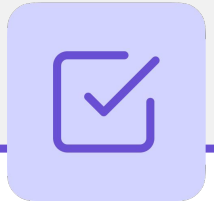
IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Форматирование чисел

Форматирование чисел



Метод `format()` и f-строки предоставляют гибкие возможности для форматирования чисел. Эти два метода работают схожим образом, но f-строки более удобны и лаконичны.

Пример

Ограничение количества знаков после запятой: Используйте спецификатор `:.nf`, где `n` — это количество знаков после запятой, `f` — спецификатор для `float`.

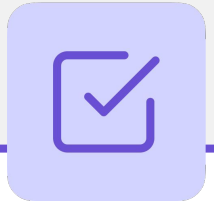
```
pi = 3.14159
# f-строки
text_fstring = f"Pi rounded to 2 decimal places is {pi:.2f}"
# Метод format()
text_format1 = "Pi rounded to 2 decimal places is {:.2f}".format(pi)
text_format2 = "Pi rounded to 2 decimal places is {0:.2f}".format(pi)
text_format3 = "Pi rounded to 2 decimal places is {num:.2f}".format(num=pi)
print(text_fstring)
print(text_format1)
print(text_format2)
print(text_format3)
```

Пример

Используйте спецификатор `:,` чтобы добавить разделители тысяч в больших числах.

```
large_number = 1234567890
# f-строки
text_fstring = f"The number with thousand separators: {large_number:,}"
# Метод format()
text_format = "The number with thousand separators: {:,}".format(large_number)
print(text_fstring)
print(text_format)
```

Выравнивание и ширина поля



И в методе `format()`, и в f-строках можно задавать выравнивание текста и чисел, а также ширину поля для их отображения. Это полезно при работе с форматированным выводом, например, в таблицах или структурированных данных.

Основные спецификаторы

- `>` — выравнивание по правому краю.
- `<` — выравнивание по левому краю.
- `^` — выравнивание по центру.

Число после символа выравнивания задаёт минимальную ширину поля, которая будет выделена для значения.

Выравнивание по правому краю

Синтаксис

```
# f-строки
text_fstring =
f"start_{'text':>10}_end"

# Метод format()
text_format = "start_{:>10}_end"
print(text_fstring)
print(text_format.format("text"))
```

Пояснения

Чтобы выравнять значение по правому краю, используйте символ >.

Выравнивание по левому краю

Синтаксис

```
# f-строки
text_fstring =
f"start_{'text':<10}_end"

# Метод format()
text_format = "start_{:<10}_end"
print(text_fstring)
print(text_format.format("text"))
```

Пояснение

Для выравнивания по левому краю используйте символ <.

Выравнивание по центру

Синтаксис

```
# f-строки
text_fstring =
f"start_{'text':^10}_end"

# Метод format()
text_format = "start_{:^10}_end"
print(text_fstring)
print(text_format.format("text"))
```

Пояснения

Для выравнивания по центру используйте символ ^.

Задание минимальной ширины поля

- Можно задать только минимальную ширину для чисел, добавив число после символа двоеточия.
- Для чисел выравнивание по умолчанию будет по правому краю.
- Для строк выравнивание по умолчанию будет по левому краю.

Пример

```python
number = 40

text = 'hi'

# f-строки

text_fstring = f"start_{number:5}_end"

# Метод format()

text_format = "start_{:5}_end"

print(text_fstring)

print(text_format.format(text))
```

Выравнивание чисел с заполнением

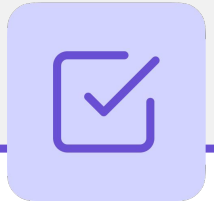
- Python позволяет выравнивать строки и числа не только с помощью пробелов, но и с заполнением другими символами, например, нулями или любыми другими символами, которые вы укажете.

- Для этого в f-строках можно указать символ заполнения перед символом выравнивания (`<`, `>`, `^`) или до значения.

Пример

```
# Заполнение нулями

number = 40

text = f"{number:0>5}"

print(text)


# Заполнение нижним подчеркиванием

text = f"{'Python':_^10}"

print(text)
```

Методы выравнивания строк



В Python для выравнивания строк существуют три метода (`center`, `ljust`, `rjust`), которые помогают располагать строки по центру, слева или справа с использованием заданной ширины и, при необходимости, с заполнением оставшегося пространства символами.

Методы выравнивания строк

- `str.ljust(width[, fillchar])` — выравнивание строки по левому краю.
- `str.rjust(width[, fillchar])` — выравнивание строки по правому краю.
- `str.center(width[, fillchar])` — выравнивание строки по центру.

Выравнивание

Синтаксис

```
str.rjust(width[, fillchar])
```

Разбор

- `width` — минимальная ширина строки.
- `fillchar` (опционально) — символ для заполнения свободного пространства (по умолчанию — пробел).

Примеры использования

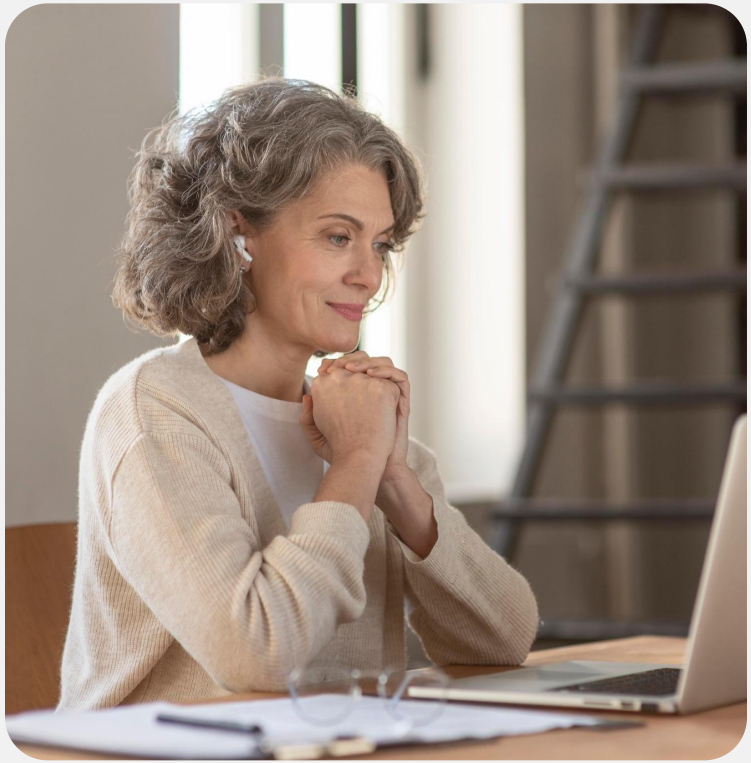
```
text = "Python"


#           ljust():           выравнивание           по           левому           краю
print(text.ljust(15))
print(text.ljust(15, '-'))

#           rjust():           выравнивание           по           правому           краю
print(text.rjust(15))
print(text.rjust(15, '-'))

#               center():               выравнивание               по               центру
print(text.center(15))
print(text.center(15, '-'))
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

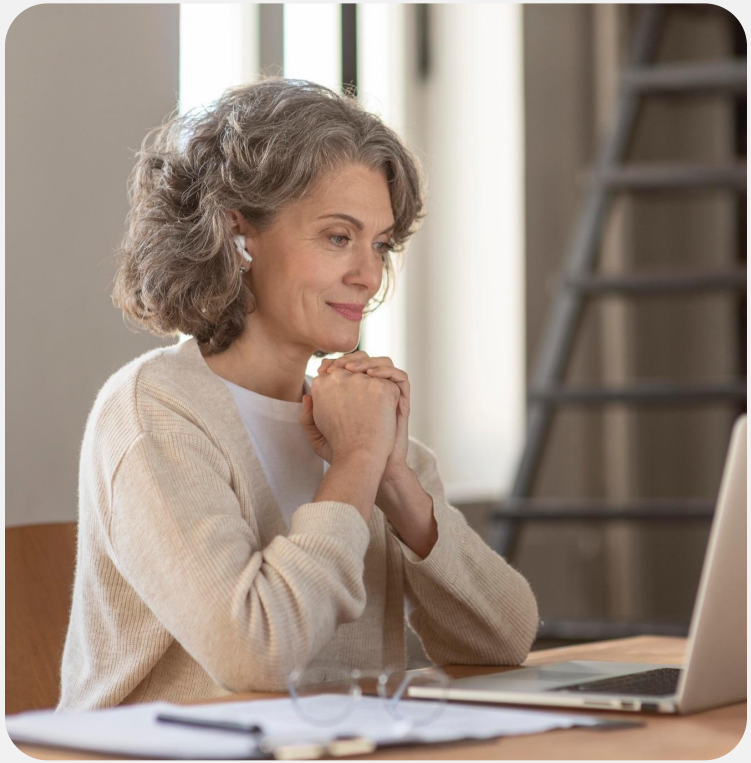


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
number = 1234.5678

print(f"Formatted number: {number:.2f}")
```

a. "Formatted number: 1234.5678"
b. "Formatted number: 1234.57"
c. "Formatted number: 1234.56"
d. "Formatted number: 1234.56f"

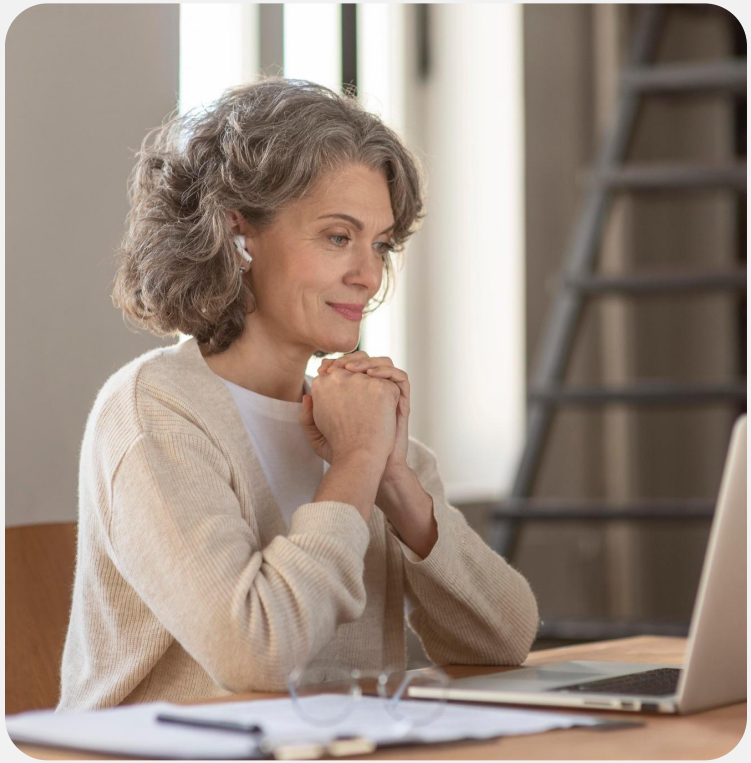


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
number = 1234.5678

print(f"Formatted number: {number:.2f}")
```

a. "Formatted number: 1234.5678"
b. "Formatted number: 1234.57"
c. "Formatted number: 1234.56"
d. "Formatted number: 1234.56f"

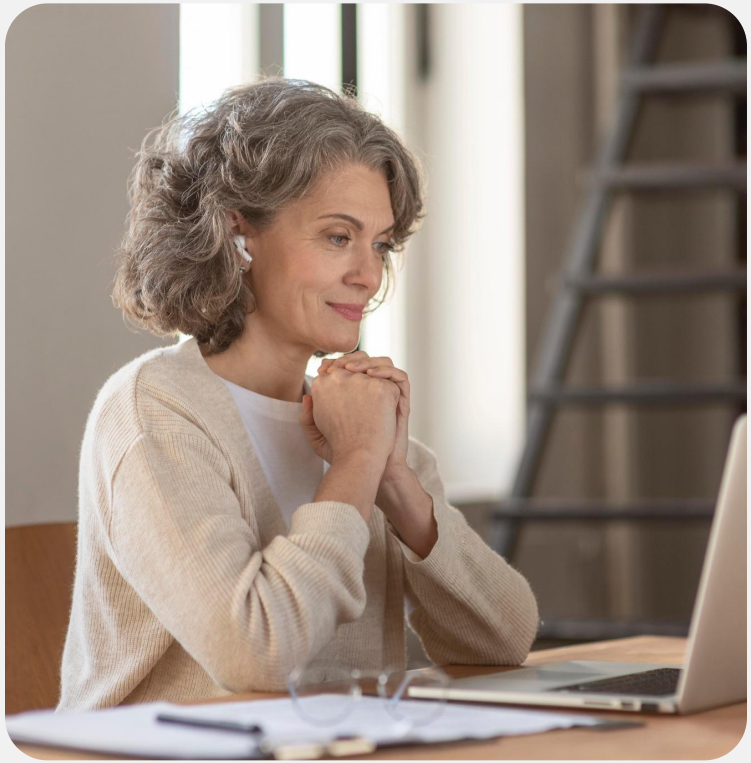


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
large_number = 9876543210

print(f"The number is:
{large_number:,}")
```

a. "The number is: 9876,543,210"
b. "The number is: 9,876543210"
c. "The number is: 9,876,543,210"
d. "The number is: 9876543210,"

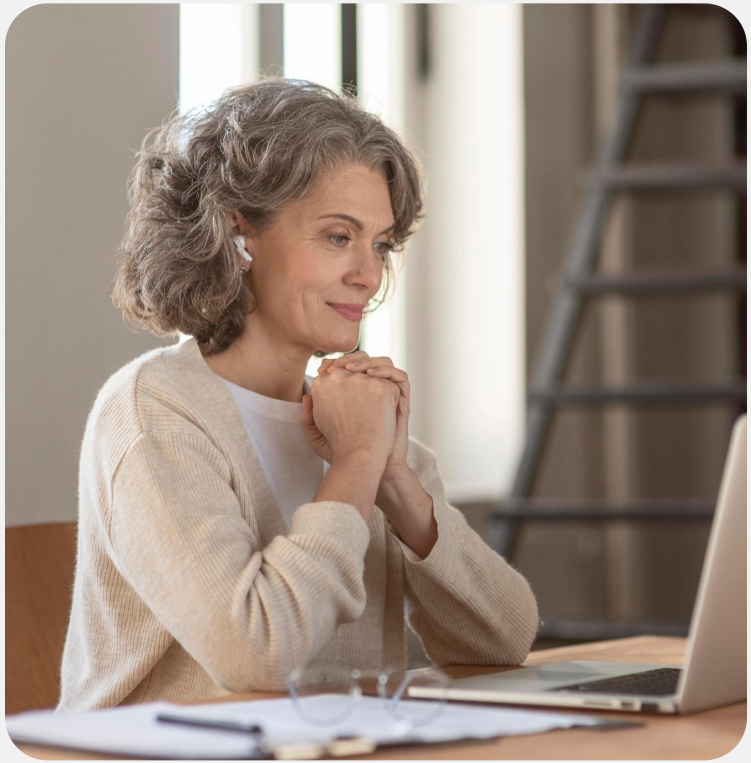


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
large_number = 9876543210

print(f"The number is:
{large_number:,}")
```

a. "The number is: 9876,543,210"
b. "The number is: 9,876543210"
c. "The number is: 9,876,543,210"
d. "The number is: 9876543210,"

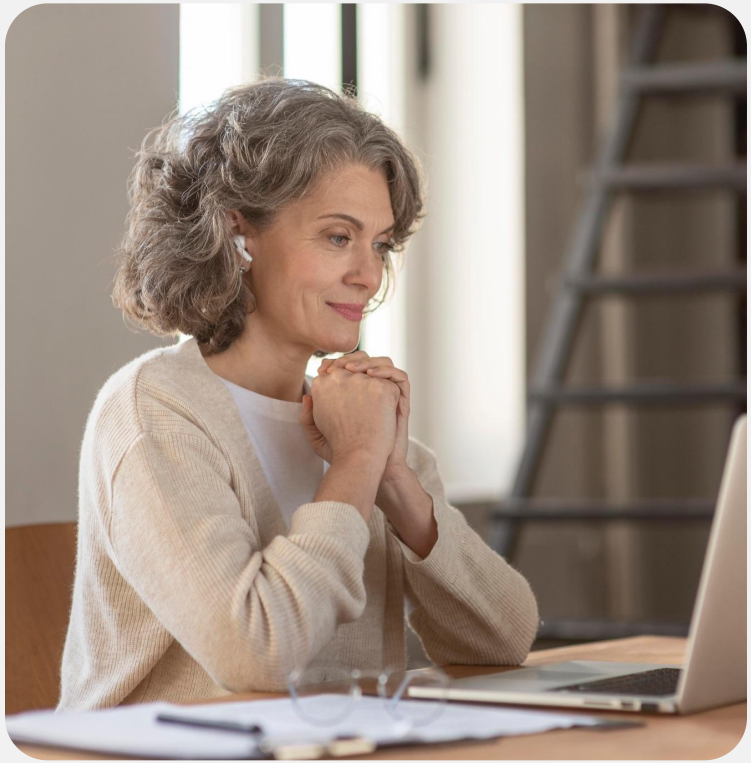


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = f"{'Python':_^10}"

print(text)
```

a. `"Python____"`
b. `"__Python__"`
c. `"Python "`
d. `Ошибка`



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
text = f"{'Python':_^10}"

print(text)
```

a. `"Python____"`
b. `"__Python__"`
c. `"Python "`
d. Ошибка

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКАЯ РАБОТА

1. Счетчик слов

Напишите программу, которая обрабатывает строку и выводит её, добавив к каждому слову его порядковый номер, выравнивая текст по левому краю с длиной в 15 символов. Слова выводите с большой буквы.

Пример вывода:

```
Введите строку: Hello world Python is great
1. Hello
2. World
3. Python
4. Is
5. Great
```

2. Формат даты

Напишите программу, которая принимает дату в виде числа, месяца и года, а затем выводит её в формате "dd/mm/yyyy", где день и месяц всегда состоят из двух цифра.

Пример вывода:

```
Введите день: 3
Введите месяц: 7
Введите год: 2024
Дата: 03/07/2024
```

IT CAREER HUB

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. Сумма положительных чисел

Напишите программу, которая обрабатывает список `float` чисел, вычисляет сумму положительных чисел. Результат необходимо вывести в формате денежной суммы (до 2 символов после запятой), добавляя символ валюты "$" вначале
и разделяя тысячи запятой.

Данные:

```
numbers = [1245.4435, -302.1403, 87459.99, -520.8265, 1450.001]
```

Пример вывода:

```
Сумма положительных чисел: $90,155.43
```

Домашнее задание

2. Счета клиентов

Напишите программу, которая принимает список строк в формате `"name age balance"` и выводит информацию о каждом человеке с отформатированными данными: имя — 10 символов, возраст — 3 символа, баланс — 10 символов с двумя знаками после запятой. Используйте заранее подготовленный список строк.

Данные:

```
data_list = [
    "John 23 12345.678",
    "Alice 30 200.50",
    "Bob 35 15000.3",
    "Charlie 40 500.75"
]
```

Пример вывода:

```
Имя: John       | Возраст:  23 | Баланс:  12345.68
Имя: Alice      | Возраст:  30 | Баланс:    200.50
Имя: Bob        | Возраст:  35 | Баланс:  15000.30
Имя: Charlie    | Возраст:  40 | Баланс:    500.75
```

Заключение