

# Python

# Strings

# Defining Strings in Python

- three ways:

```
>>> x = 'Mozzarella'      # single quotes  
>>>  
>>> x = "Mozzarella"    # double quotes  
>>>  
>>> x = """Mozzrella""" # triple quotes
```

- use escape (\) for special characters

```
>>> print("\a")          # system bell
```

# Character Encoding

- ASCII: each character is one byte
- (extended) ASCII: up to (256) 128
- not enough for many alphabets
- Unicode: up to 4 bytes per character
- three encoding schemes:
  1. UTF-32: use 4 bytes
  2. UTF-16: use 2 bytes
  3. UTF-8: variable length encoding

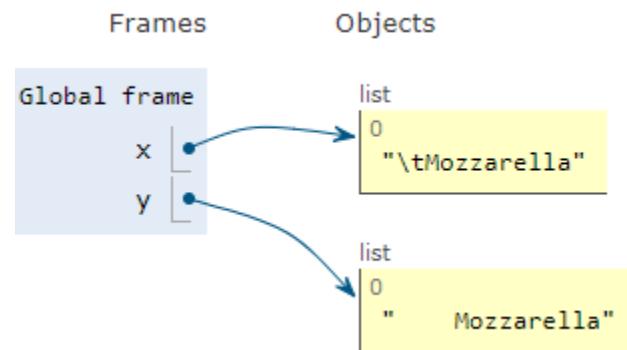
# Character Escape Codes

character	description
\	newline continue
\\	backslash
\a	system bell
\b	backspace
\e	escape
\t	horizontal tab
\v	vertical tab
\"	double quotes

- raw strings: ignore escape codes

```
>>> x = r'\tMozzarella'
```

```
>>> y = '\tMozzarella'
```



# Escape Sequences

- Use “\” (backslash) for special characters:

```
>>> x = 'Mozzarella\'s' # single quotes  
Mozzarella's  
>>> x = "Mozza\trella" # tab  
Mozza    rella  
>>> x = "Mo\"zz\"rella" # double quotes  
Mo"zz"rella  
>>> x = "D:\\bu\\python"  
D:\\bu\\python
```

- for dir/file names, use this:

```
>>> import os  
>>> x = os.path.join("D:", "bu", "python")  
D:\\bu\\python
```

# String Operations

- concatenation

```
>>> "Mozzarella" + "Cheddar"
```

```
'MozzarellaCheddar'
```

- repetition

```
>>> "Mozzarella" * 3
```

```
'MozzarellaMozzarellaMozzarella'
```

- indexing

```
>>> "Mozzarella"[0]
```

```
'M'
```

- slicing

```
>>> "Mozzarella"[0 : 4]
```

```
'Mozz'
```

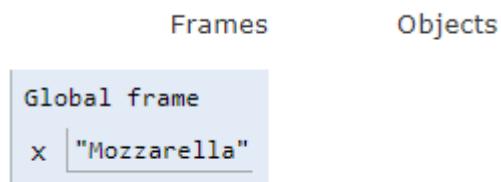
# String Immutability

- once created, cannot be changed

```
>>> x = "Mozzarella"
```

```
>>> id(x)
```

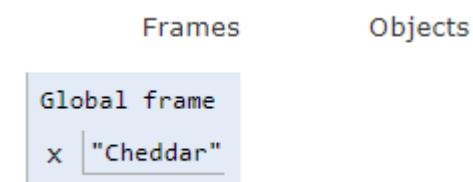
139685060402176



```
>>> x = "Cheddar"
```

```
>>> id(x)
```

139760688281904



# String as an Object

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozarella"
```

```
>>> y = x.index('r')
```

```
>>> y
```

```
>>> 5
```

- string is an object
- what are string attributes?

variable name	delimiter	attribute	arguments
X	.	index	('r')

# String as an Object

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"      # double quotes  
>>> x = 'Mozzarella'    # single quotes  
>>> x = """Mozzarella"""\n# triple quotes  
>>> y = x.index('r')  
>>> y  
>>> 5
```

- string is an object
- what are string attributes?

variable name	delimiter	attribute	arguments
X	.	index	('r')

# Slicing

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

>>> x = "Mozzarella"

- slice operator:

variable		start		one past end	
x	[	i	:	j	]

# Indexing for String Slicing

- each element is identified by its position
- slice – elements between two positions

```
>>> x = "Mozzarella"
```

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e			a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- indexing starts at 0, ends at (length – 1)
- can refer to slices but positive or negative indices

# Slicing with a Single Index

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

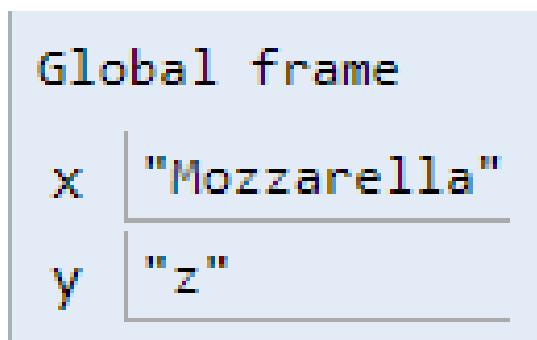
```
>>> y = x[2]
```

```
>>> y
```

```
>>> z
```

Frames

Objects



# Slicing with Positive Indices

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[3 : 7]
```

```
>>> y
```

```
>>> zare
```

Frames

Objects

Global frame	
x	"Mozzarella"
y	"zare"

# Slicing w. Negative Indices

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[-7:-2]
```

```
>>> y
```

```
>>> zarel
```

Frames

Objects

```
Global frame
x  "Mozzarella"
y  "zarel"
```

# Slicing with Positive and Negative Indices

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"  
>>> y = x[3 : -2]  
>>> y  
>>> zarel
```

Frames

Objects

Global frame	
x	"Mozzarella"
y	"zarel"

# Extended Slicing

- [start: finish: count]

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> # extract every third letter  
>>> x = "Mozzarella"  
>>> y = x[0 : 10 : 3]  
>>> y  
>>> Mzea
```



# Slice Replacing

- strings are immutable
- cannot change substrings without creating new strings!!!

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
Console
Python 1 IP: Kernel 1
>>>
>>>
>>> x = "Mozzarella"
>>> x[2:5] = 'xyz'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```

The screenshot shows a Jupyter Notebook console window. The title bar says "Console". There are two tabs: "Python 1" and "IP: Kernel 1". The Python tab is active. The code entered is: `>>> x = "Mozzarella"`, `>>> x[2:5] = 'xyz'`. This results in a traceback error: `Traceback (most recent call last):`,  `File "<stdin>", line 1, in <module>`, `TypeError: 'str' object does not support item assignment`. The bottom status bar shows "Encoding: UTF-8-GUESSED", "Line: 5", "Column: 1", and "Memory: 42 %".

# Substring Replacement

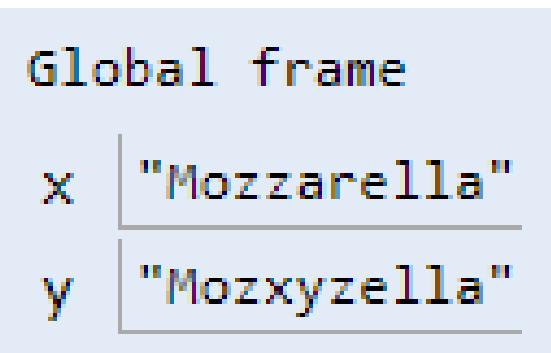
- can create a new string

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"  
>>> y = x.replace("zar", "xyz")  
>>> y  
>>> Mozxyzella
```

Frames

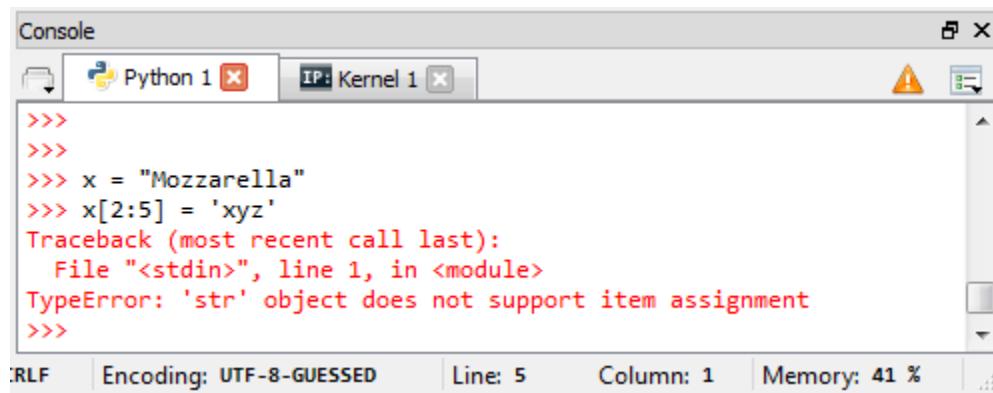
Objects



# Valid vs. Invalid Slicing

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

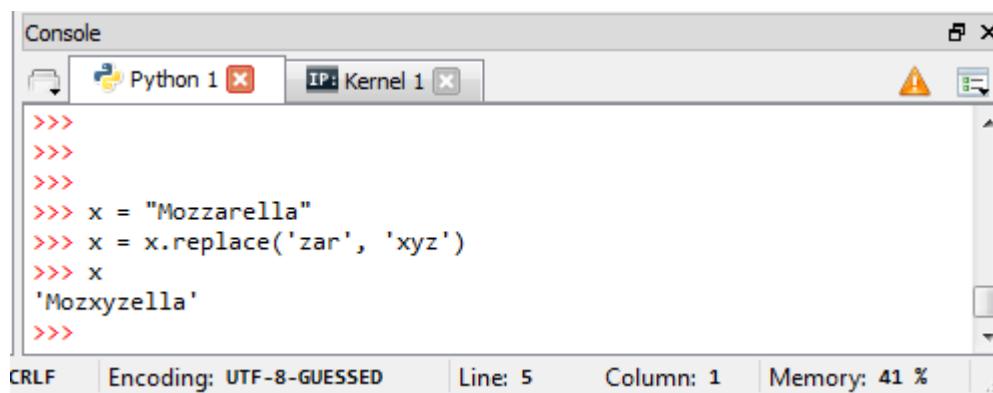
- why is this illegal?



```
Console
Python 1 IP: Kernel 1
>>>
>>>
>>> x = "Mozzarella"
>>> x[2:5] = 'xyz'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>>
```

CRLF Encoding: UTF-8-GUESSED Line: 5 Column: 1 Memory: 41 %

- why is this legal?



```
Console
Python 1 IP: Kernel 1
>>>
>>>
>>>
>>> x = "Mozzarella"
>>> x = x.replace('zar', 'xyz')
>>> x
'Mozxyzella'
>>>
```

CRLF Encoding: UTF-8-GUESSED Line: 5 Column: 1 Memory: 41 %

# String Reversal

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"  
>>> y = x[::-1] # idiomatic  
>>> z = "".join(reversed(x)) # slower
```

Frames

Objects

Global frame	
x	"Mozzarella"
y	"allerazzom"
z	"allerazzom"

# Slicing: Quiz

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- what is the result of these operations?

```
>>> x = "Mozzarella"
```

```
>>> y = x[-1]
```

```
>>> z = x[2 : 6]
```

```
>>> w = x[-7 : -3]
```

```
>>> v = x[ : ]
```

# Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	o	z	z	a	r	e	l	l	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

>>> x = "Mozzarella"

Frames

Objects

```
Global frame  
x  "Mozzarella"
```

# Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[-1]
```

Frames

Objects

Global frame

x "Mozzarella"

y "a"

# Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[-1]
```

```
>>> z = x[2 : 6]
```

Frames

Objects

Global frame

x "Mozzarella"

y "a"

z "zzar"

# Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[-1]
```

```
>>> z = x[2 : 6]
```

```
>>> w = x[-7 : -3]
```

Frames

Objects

Global frame	
x	"Mozzarella"
y	"a"
z	"zzar"
w	"zare"

# Slicing: Quiz (cont'd)

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"
```

```
>>> y = x[ -1]
```

```
>>> z = x[2 : 6]
```

```
>>> w = x[ -7 : -3]
```

```
>>> v = x[ : ]
```

Frames

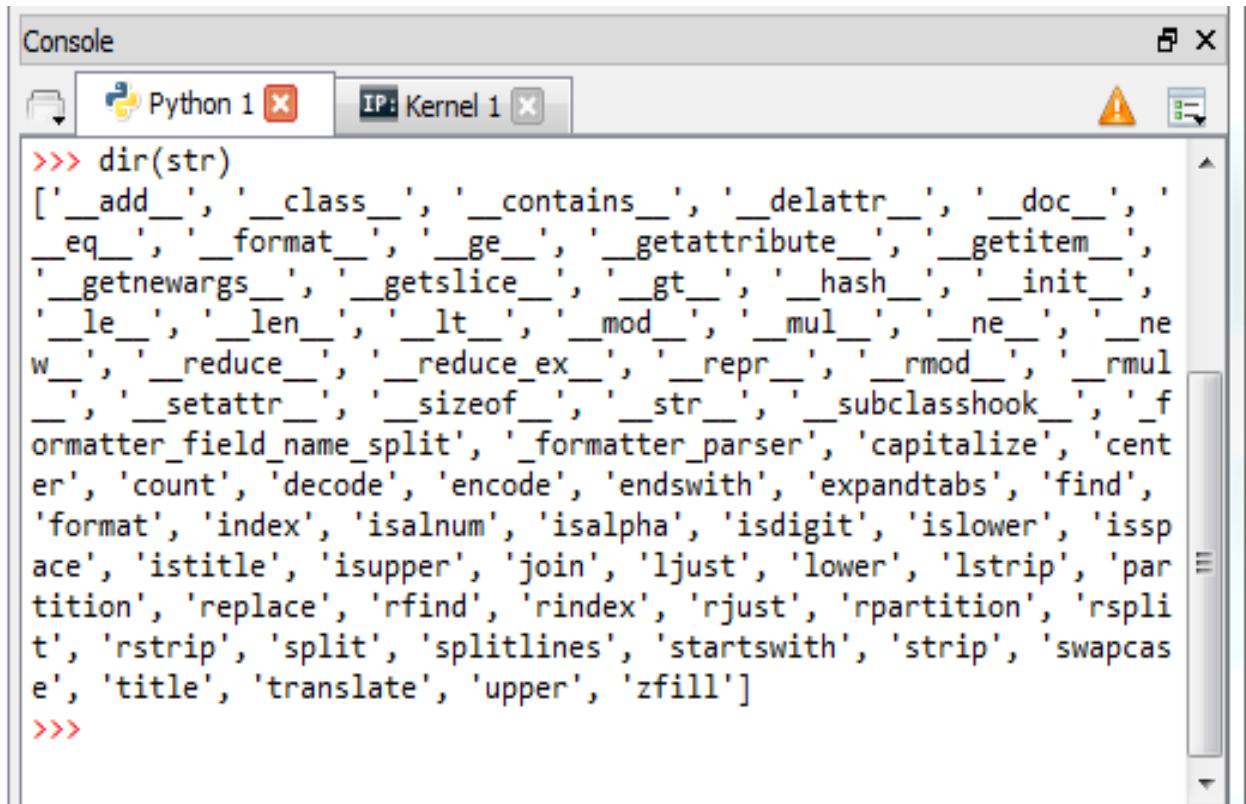
Objects

Global frame	
x	"Mozarella"
y	"a"
z	"zare"
w	"zare"
v	"Mozarella"

- example of a shallow copy

# String Attributes

- use `dir(str)` command



The screenshot shows a Jupyter Notebook interface with a "Console" tab selected. The Python 1 kernel is active. The code cell contains the command `>>> dir(str)`. The output is a list of string methods, starting with `'__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'formatter_field_name_split', 'formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']`. The IP: Kernel 1 tab is also visible.

- names starting with `__` refer to “local” class names/methods

# String Comparisons

0	1	2	3	4	5	6	7	8	9
M	O	Z	Z	a	r	e	I	I	a
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> x = "Mozzarella"  
>>> y = x.upper()  
>>> condition_1 = (x[4] == y[9])  
>>> condition_2 = (x[4] == y[9].lower())  
>>> condition_3 = (x[0 : 2] <= y[0 : 2])  
>>> condition_4 = (x[0 : 2] > y[0 : 2])
```

	Frames	Objects
Global frame	x y condition_1 condition_2 condition_3 condition_4	"Mozzarella" "MOZZARELLA" False True False True

# Review Problems

# Interview Problem

- what is negative index?

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon ('?') and settings icon ('⚙️') on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is visible above the main content area. The main content area contains the following text:

Exercise 51098 —

Write an expression that evaluates to True if and only if `s` refers to the str "end".

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and gear icon on the right. Below these is the title 'Exercise 51099 —'. Underneath the title are two tabs: 'WORK AREA' (which is selected) and 'SOLUTIONS'. A green button labeled 'Content Support' is located below the tabs. The main content area contains the following text:

Write an **expression** that evaluates to True if the str associated with s1 is **greater than** the str associated with s2.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help/gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Write an **expression** that evaluates to True if the variable `last_name` refers to a str that is **greater** than the str "Dexter".

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue and red, and a help/gear icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Write an **expression** that evaluates to True if and only if the variable `s` does **not** refer to the str "end".

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button with a gear icon, and a help icon (question mark) and settings icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Write an **expression** whose value is the character at index 3 of the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below the navigation is the title 'Exercise 51761 —'. Underneath the title are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located below the tabs. The main area contains the following text: 'Write an expression whose value is the last character in the str associated with s.'

WORK AREA      SOLUTIONS

Content Support

Write an expression whose value is the last character in the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button, a help icon (a question mark), and a settings gear icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located in the top right corner of the main content area. The main content area contains a question: "Write an **expression** whose value is the str that consists of the **second to last character** of the str associated with s."

Exercise 51762 —

WORK AREA    SOLUTIONS

Content Support

Write an **expression** whose value is the str that consists of the **second to last character** of the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue, and a help icon ('?') and settings icon ('⚙️'). Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located in the top right corner of the main content area. The main content area contains the following text:

Exercise 51763 –

Write an **expression** whose value is the str that consists of the **third to last character** of the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, a 'Workbench' button in blue, and a help/gear icon. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

Write an **expression** whose value is the str that consists of the **first four characters** of the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' in green rounded rectangles, followed by a 'Workbench' button in blue, and a help icon ('?') and settings icon ('⚙️') in green. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A 'Content Support' button is located above a large text area. The main text area contains the following question:

**Write an expression whose value is the str that consists of the second through fifth characters of the str associated with s.**

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help/gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located near the bottom left of the main content area. The main content area contains the following text:

Exercise 51771 –

Write an **expression** whose value is the str consisting of all the characters (starting with the sixth) of the str associated with s.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a question mark icon and a gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Given that `s` refers to a string, write an expression that evaluates to a string that is a substring of `s` and that consists of all the characters of `s` from its start through its ninth character.

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**

# Programming Exercise

The screenshot shows a user interface for a programming exercise. At the top, there are navigation buttons: 'PREV' and 'NEXT' on the left, 'Workbench' in the center, and a help icon with a question mark and a gear icon on the right. Below these are two tabs: 'WORK AREA' and 'SOLUTIONS'. A green button labeled 'Content Support' is located above the main content area. The main content area contains the following text:

Assume that `name` is a variable of type **String** that has been assigned a value. Write an expression whose value is a **String** containing the first **character** of the value of `name`. So if the value of `name` were "**Smith**" the expression's value would be "**S**".

# **Programming Exercise**

## **Worksheet**

**this page is intentionally left blank**