Strings.

- 11.2.2.3 apply functions and string processing methods;
- 11.2.2.1 perform access to the elements of strings, lists, tuples;
- 11.2.2.2 use slicers to process the string;
- 11.2.3.6 determine the difference between different data structures.

String

We already saw the **Set collection** in the previous lesson, which can store more than one value. Today we are getting acquainted with the second collection - **String**. And although we have often used the string type, this is the first time we will consider a **string as a collection of symbols**.

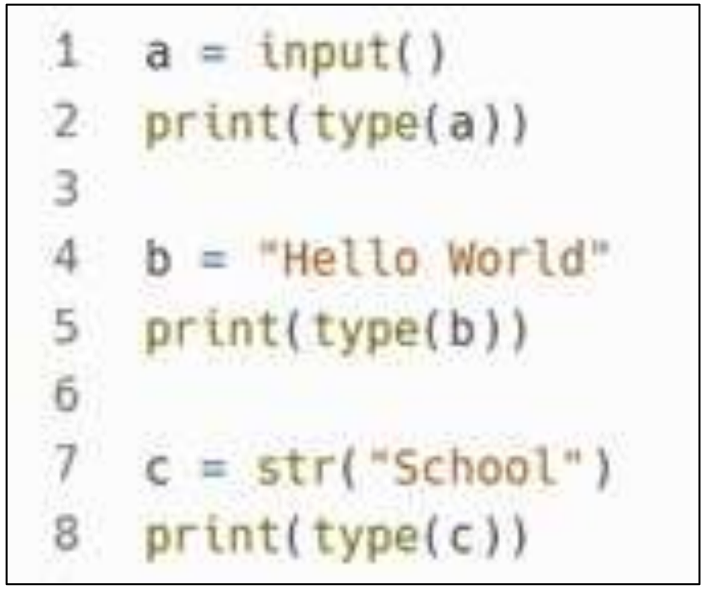
***String** is a complex data type that stores a sequence of characters.*

String data structure features:

- Strings are an **ordered** data structure. Each element has its own index. Indexing starts from 0 (zero).
- Strings are an **iterable** data structure. You can iterate over the elements of strings in two ways, by characters, and by indices.
- Each element of the string is represented by one **character**.
- Strings are an **immutable** data structure. You cannot replace a character in a string without conversions.

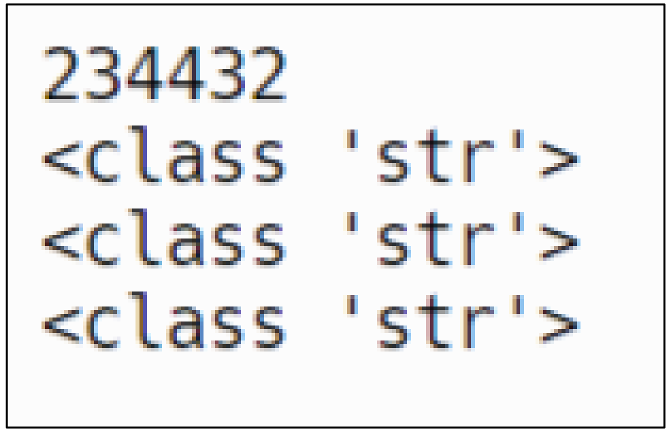
Compare with Sets

We get the string when we enclose characters in quotation marks, for example, "text", when we use input() when we use the conversion function str ().



```python
a = input()
print(type(a))

b = "Hello World"
print(type(b))

c = str("School")
print(type(c))
```

Output:

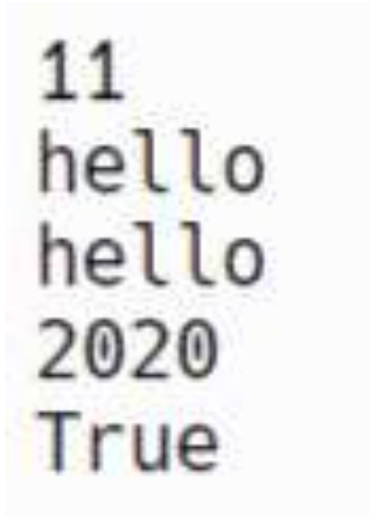


```
234432
<class 'str'>
<class 'str'>
<class 'str'>
```

We can already measure the length of a string using the **len()** function and determine the **occurrence of one string** within another using the in operation.

```
fixed_line = "programming"
print(len(fixed_line)) # print line length

word = input () # input string
print(word) # line output

number = 2020
line = str(number) # convert number to string
print(line)

print("program" in fixed_line) # checking the occurrence of one line in another
```

What the program will output?



```
11
hello
hello
2020
True
```

Indexing

Unlike sets, which contain unordered elements, the sequence of characters in **a string has its own order**, and **each character has its own address in the string - an index by which it can be accessed.**

	0	1	2	3	4	5	6	7	8	9	10
s =	**P**	**R**	**O**	**G**	**R**	**A**	**M**	**M**	**I**	**N**	**G**

You can refer to each character by its index. To do this, use the following entry:
s[1] - refer to the character with index 1 in string s.

```
s = "programming"
print(s[1]) # print the letter "r"
print(s[6] + s[5] + s[0]) # output of the word "map"
```

What the program will output?

```
s = "programming"
print(s[1])

print(s[6] + s[5] + s[0])
```

Output:

```
r

map
```

Negative indexes

	0	1	2	3	4	5	6	7	8	9	10
s =	**P**	**R**	**O**	**G**	**R**	**A**	**M**	**M**	**I**	**N**	**G**
	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

When accessing an index that does not exist, we will get an error.

```
s = "programming"
print(s[12]) # IndexError: string index out of range # Index error: row index is
out of range
```

If we want to correct one letter in a word, for example,

```
s = "pragramming"
s[2] = "o" # TypeError: 'str' object does not support item assignment
```

The Python interpreter throws an error - it means that it is impossible to change a single character of the string, that is, the string is an **immutable data type** in Python.

What the program will output?

```
s = "programming"
print(s[-3])

print(s[-4] + s[-3] + s[-10])
```

Output:

```
i


mir
```

Iterating over the elements of a string.

Since a string is a collection that consists of character elements, we can iterate over each element of the string as we iterate over sets. To iterate over, we will use a for a loop.

```
s = "programming"
for item in s: # iterator item will iterate over each character of the string
print(item, end=" ") # output each character of the string separated by a space
```

What the program will output?

Slices

Slices have their own syntax.
The slice has three parameters, the starting element index START, the ending element index STOP (not including the ending element), and the STEP:

```
string[START:STOP:STEP]
```

Several examples with different parameters:

```
[:]/[::] # all elements,
[::2] # odd elements in order,
[1::2] # even elements in order,
[::-1] # all elements in reverse order,
[5:] # all elements starting from the sixth element,
[:5] # all elements before the sixth element,
[-2:1:-1] # all elements from the penultimate to the second in reverse order
(in all cases of sampling from a larger index to a smaller one, you must
specify a step!).
```

What the program will output?

```
s = "programming"
slice = s[:]
print(slice)          "programming"

slice = s[:5]
print(slice)          "progr"

slice = s[4:]
print(slice)          "ramming"

slice = s[::2]
print(slice)          "pormig"

slice = s[1::2]
print(slice)          "rgamn"

slice = s[::-1]
print(slice)          "gnimmargorp"
```

What the program will output?

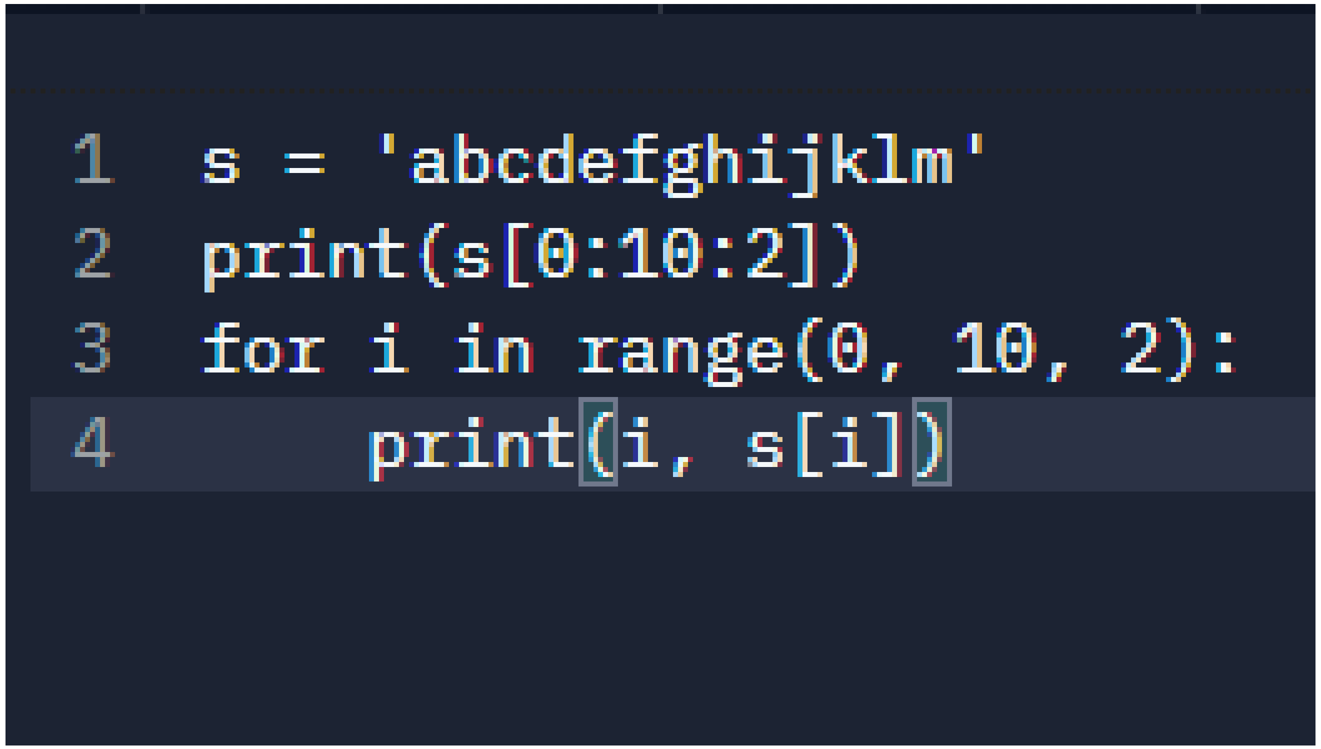
```
s = "programming"
slice = s[2:9:3]     "oai"
print(slice)

slice = s[10:0:-4]   "gmo"
print(slice)

slice = s[8:15]      "ing"
print(slice)

slice = s[12::]       output empty string
print(slice)

slice = s[:-5]       "progra"
print(slice)

slice = s[-8:-4]     "gram"
print(slice)
```

Slices

Slice can be used as a value of iterator.

```
s = "programming"
for i in s[::2]: #iterate over all elements that have even indices in string s
print(i, end=" ") # output "p o r m i g"
```

Slices



```
s = 'abcdefghijklm'
print(s[0:10:2])
for i in range(0, 10, 2):
    print(i, s[i])
```

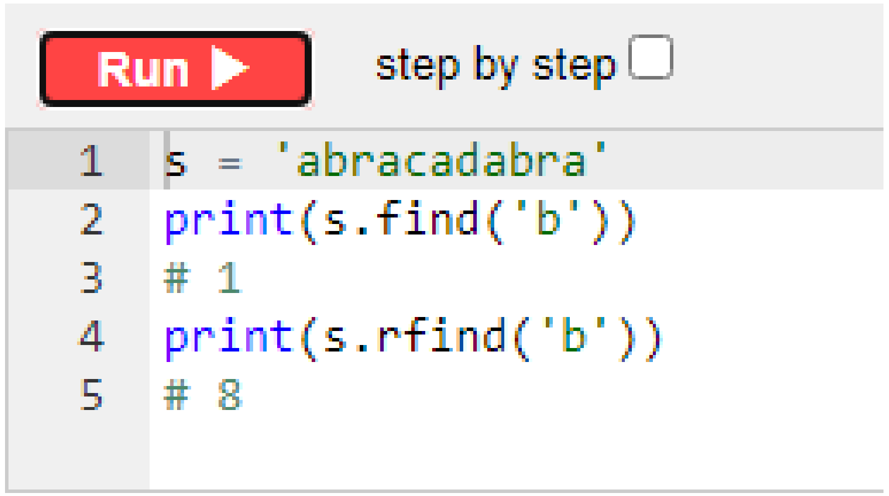
If you specify a slice with three parameters S[a:b:d], the third parameter specifies the **step**, same as for function range().

String methods: find() and rfind()

- A method is a function that is bound to the object. When the method is called, the method is applied to the object and does some computations related to it.
- Methods are invoked as object_name.method_name(arguments).

For example, in s.find("e") the string method find() is applied to the string s with one argument "e".

- Method find() searches a substring, passed as an argument, inside the string on which it's called. The function returns the index of the first occurrence of the substring. If the substring is not found, the method returns -1.



```
s = 'abracadabra'
print(s.find('b'))
# 1
print(s.rfind('b'))
# 8
```

What will the program output?

```
s = 'Hello, world!’
print(s.find('e’))
print(s.find('ll’))
print(s.rfind('l’))
print(s.find('l'))
```

```
1
2
10
2
```

String methods: replace(old, new)

- Method replace() replaces all occurrences of a given substring with another one.
- Syntax: s.replace(old, new) takes the string S and replaces all occurrences of substring old with the substring new. Example:

What will the program output?

```
s = 'life is life’
print(s.replace('life','love'))
```

```
Love is love
```

replace(old, new, count)

- One can pass the third argument count, like this: s.replace(old, new, count).
- It makes replace() to replace only first count occurrences and then stop.

What will the program output?

```
s = 'life is life’
print(s.replace('life','love’),1)
```

Love is life

String methods: count()

- This method counts the number of occurrences of one string within another string. The simplest form is this one: s.count(substring).
- Only non-overlapping occurrences are taken into account:
- If you specify three parameters s.count(substring, left, right), the count is performed within the slice s[left:right].

What will the program output?

```
s = 'life is life’
print(s.count(' '))
```

```
2
```

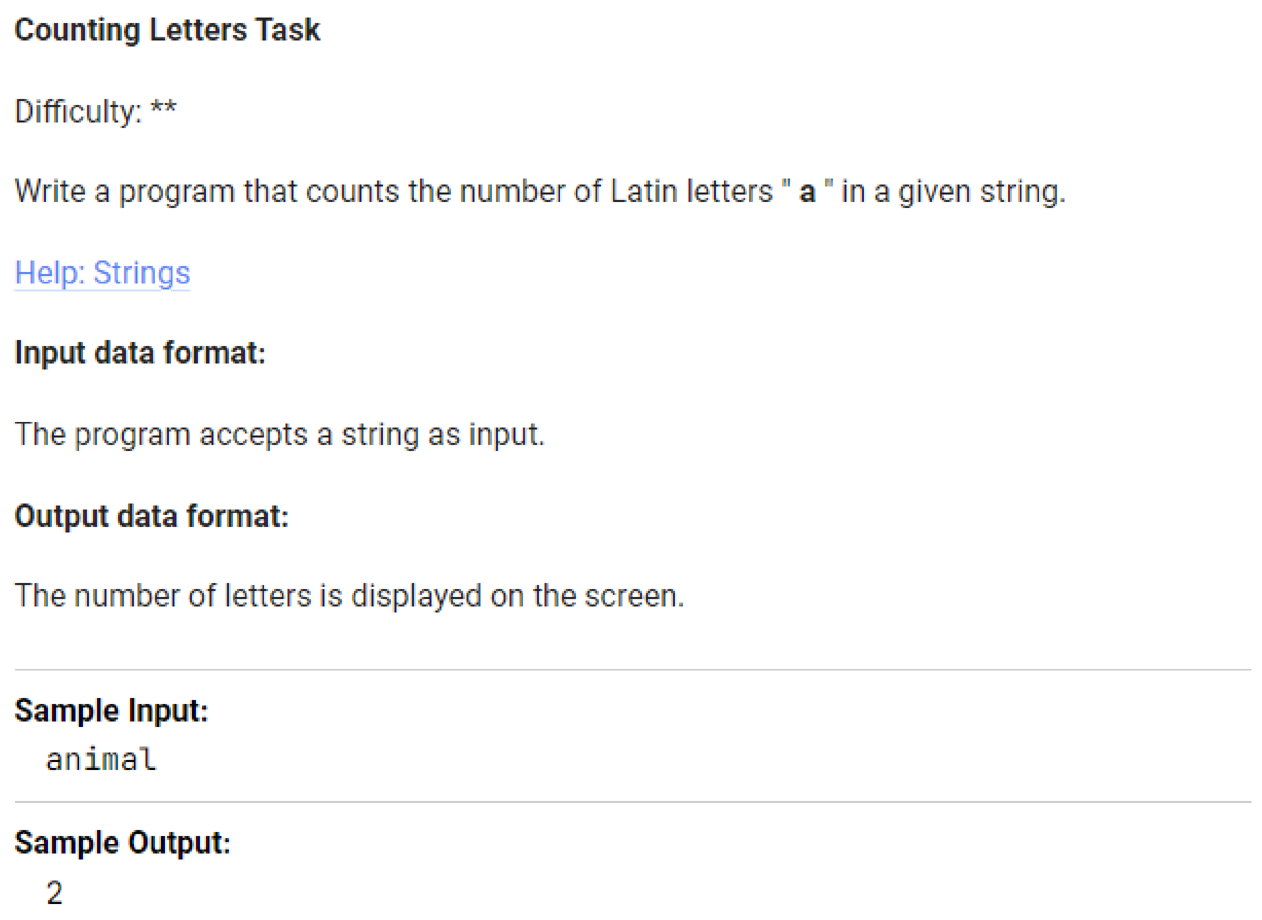
Task 1 - You are given a string. Using slices define

- In the first line, print the third character of this string.
- In the second line, print the second to last character of this string.
- In the third line, print the first five characters of this string.
- In the fourth line, print all but the last two characters of this string.
- In the fifth line, print all the characters of this string with even indices (remember indexing starts at 0, so the characters are displayed starting with the first).
- In the sixth line, print all the characters of this string with odd indices (i.e. starting with the second character in the string).
- In the seventh line, print all the characters of the string in reverse order.
- In the eighth line, print every second character of the string in reverse order, starting from the last one.
- In the ninth line, print the length of the given string.

Tasks 2,3,4

- Given a string. Cut it into two "equal" parts (If the length of the string is odd, place the center character in the first string, so that the first string contains one more characther than the second). Now print a new string on a single row with the first and second halfs interchanged (second half first and the first half second) Don't use the statement if in this task.
- Given a string consisting of exactly two words separated by a space. Print a new string with the first and second word positions swapped (the second word is printed first).
- Given a string that may or may not contain a letter of interest. Print the index location of the first and last appearance of f. If the letter f occurs only once, then output its index. If the letter f does not occur, then do not print anything.

Let's solve together!



Counting Letters Task

Difficulty: **

Write a program that counts the number of Latin letters " **a** " in a given string.

Help: Strings

Input data format:

The program accepts a string as input.

Output data format:

The number of letters is displayed on the screen.

Sample Input:

```
animal
```

Sample Output:

```
2
```

```
s=str(input())
c=0
for x in s:
    if x=="a":
        c+=1
print(c)
```

Tasks

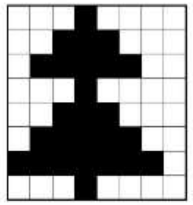
String Indexing
https://stepik.org/lesson/449974/step/1?unit=440357

String Slicing
https://stepik.org/lesson/449976/step/1?unit=440359

Task Indexing 0,25 points

Picture Inversion Quest

Difficulty: **



```
00010000
00111000
01111100
00010000
00111000
01111100
11111110
00010000
```

A black and white image in a computer is represented as a binary code (sequences 0 and 1). Write a program that inverts the bits in a character string: replaces all zeros with ones in it, and vice versa.

Help: Strings

Input data format:

At the input, the program accepts a string - a bit sequence of the picture.

Output data format:

A string with inverted bits is displayed on the screen.

Sample Input:

```
00110010
```

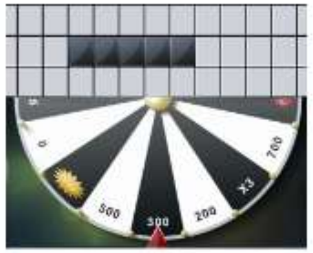
Sample Output:

```
11001101
```

Task Indexing 0,25 points

The problem "There is such a letter!"

Difficulty: **



In the TV show "Field of Miracles" the player names a letter, and the presenter checks if there is such a letter on the scoreboard.

Write a program that asks the user for a letter and checks if there is such a letter in a word. The position of the letter or letters (if there are several) in the word is displayed on the screen. If the word does not contain a letter, then the message "NO" is displayed.

Note: letter numbering starts with 1.

Help: Strings

Input data format:

At the entrance, it requests a string and a letter.

Output data format:

The screen displays the number of a letter in a word (or numbers of letters in a line separated by a space, if there are many) or "NO". E.g: banana -> 2 4 6

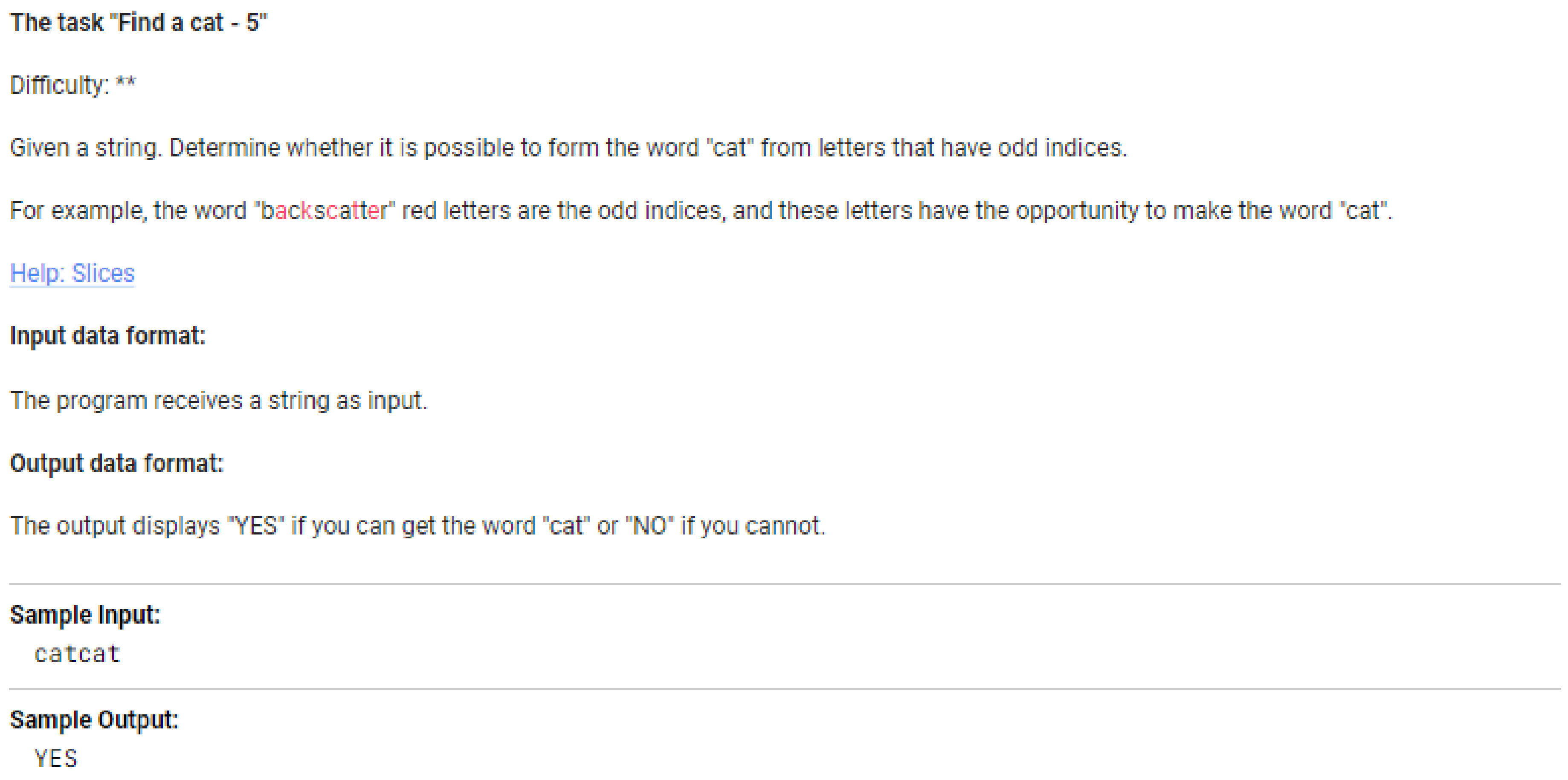
Sample Input:

```
animal
a
```

Sample Output:

```
1 5
```

Task Slices 0,25 points



The task "Find a cat - 5"

Difficulty: **

Given a string. Determine whether it is possible to form the word "cat" from letters that have odd indices.

For example, the word "backscatter" red letters are the odd indices, and these letters have the opportunity to make the word "cat".

Help: Slices

Input data format:

The program receives a string as input.

Output data format:

The output displays "YES" if you can get the word "cat" or "NO" if you cannot.

Sample Input:

```
catcat
```

Sample Output:

```
YES
```

Task Slices 0,25 points

The task "Happiness in St. Petersburg"

Difficulty: ***

A ticket is called lucky in St. Petersburg if the sum of the digits of its number in even places is equal to the sum of the digits in odd places.

Write a program that determines whether a ticket is lucky in St. Petersburg.

Help: Slices

Input data format:

The number of a ticket.

Output data format:

The output is "Happy" if the condition is met or "Unhappy" if not.

Sample Input:

```
123456
```

Sample Output:

```
Unhappy
```

Reflection

- What knows?
- What remained unclear
- What is necessary to work on?

Send a sticker
to in telegram that describes the lesson)