Strings

What is a string?

A string is a sequence of characters. (Here each character has a position associated with it.)

When we write “hello world!” what computer sees is this:

h	e	l	l	o		w	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11
j	a	c	k		s	m	i	t	h		
0	1	2	3	4	5	6	7	8	9		

String Indexing

Just a string	f	o	o	b	a	r
position	0	1	2	3	4	5
string	b	o	b			
position	0	1	2			
string	a	l	i	c	e	
position	0	1	2	3	4	
string	h	e	l	l	o	
position	0	1	2	3	4	

A Word About String Indexing

- Indexing starts from 0
- Indexing ends at len() - 1

```
>>> s = 'rakesh'
>>> s[0]
'r'
>>> s[1]
'a'
>>> s[2]
'k'
>>> s[3]
'e'
>>> s[4]
's'
>>> s[5]
'h'
>>> s[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>>
```

home > ashish > Desktop > string variable declaration and indexing.py > ...

```python
# Variable declaration happens in three parts.
# First part is variable name. Second part is assignment operator.
# And third part is the variable value.

myname = 'Mahi'
mysister = 'Chulbul'

# Printing a variable
print(myname)
# Prints "Mahi"

# Indexing: Accessing a character in a string through it's position.
# Indexing of strings starts from 0.

letter0 = myname[0] # Here we are accessing letter at position 0.
print(letter0) # Prints M

letter1 = myname[1] # Here we are accessing letter at the position number 1.
print(letter1)

letter2 = myname[2]
print(letter2) # Prints h

letter3 = myname[3]
print(letter3)

char0 = mysister[0]
print(char0) # Prints C
```

First Set of Questions on Strings

1. Declare a string variable
2. Print the third letter from that string
3. Print the length of the string variable

Q: If last index of a string is 15, what is it’s length?
Answer: 16
Q: If the length is 7, what is the last index?

What is a slice and why is it needed?

J	a	c	k		S	m	i	t	h		J	u	n	i	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

With indexing: we can access a character in the string.

What if I ask you to extract just “Jack” for me?

What about just “Smith”?

What about “Junior”?

Building a slice

J	a	c	k		S	m	i	t	h		J	u	n	i	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

For Jack: starting index is: 0, ending index is: 3

So, the slice is: [0:4]

For Smith: starting index is: 5, ending index is: 9

So, the slice is: [5:10]

For Junior: starting index is: 11, ending index is: 16

So, the slice is: [11:17]

```
x = "Jack Smith Junior"
```

```
x[0:4]
```

```
'Jack'
```

```
x[5:10]
```

```
'Smith'
```

```
x[11:17]
```

```
'Junior'
```

Demonstrating Negative Step For a Slice

J	a	c	k		S	m	i	t	h		J	u	n	i	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Let us say that this time we set step = -1

Then, the traversing of the string would happen from right to left.

Now, let us say start index: 15

Now, let us say end index: 5

This will traverse the string from right to left: from index 15 to index 6 because it is exclusive of end index.

```
x[15:5:-1]
```

```
'oinuJ htim'
```

One more example: For Step > 1

J	a	c	k		S	m	i	t	h		J	u	n	i	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Now, let us say our step = 2

This time it will not traverse each index, it will increment by 2 to pick an index.

And, start index is: 0

End index is: 17 (Why 17? Because slicing is exclusive of end index.)

```
x[0:17:2]
```

```
'Jc mt uir'
```

What is a Slice?

A slice has three parts.

First part is: start index

Second part is: end index

Third part is: step (optional)

Representation: [start index: end index: step]

Default values

Start index: 0

End index: len(mylist) or len(mystr) when step is positive.

Step: 1

Negative step means direction of traversing is from right to left.

Slice (example)

j	a	c	k		s	m	i	t	h
0	1	2	3	4	5	6	7	8	9

If the string “jack smith” is given to you, then what all possible values can start index take?
Answer: 0, 1, 2, ..., 7, 8, 9

Let’s assume that step = 1.
What all possible values can end index take?
Answer: it depends on start index.

```
>>> x = 'jack smith'
>>> x[5:9]
'smit'
>>> x[5:2]
''
```

Problems on slice expansion.

- What is the expansion of [2:7]
- Start index is 2. End index is 7. Step takes the default value: 1 --> [2:7:1]
- What is the expansion of [:2]
- Start index is not given. Takes the default of 0. End index is given 2 --> [0:2:1]
- What is the expansion of [3:]
- One colon is there. Step is optional. So step takes the default value. Start index is given to be 3. End index takes the default value: len() --> [3:len():1]
- What is the expansion of [:]
- Read as: End to end. All the default values will be there --> [0:len():1]
- What is the expansion of [::-1]
- This comes in handy when you want to reverse a string.
- Since start index is not there and end index is also not there, we would assume that it would go end to end. And since step is negative we would traverse the list from right to left.

  But it is not equal to [0:len(x):-1]

String Slicing

String is a sequence of characters.

X = ‘jack smith’

x[0] # j

x[1] # a

x[2] # c

```
x = "Apple"
x[1:3] # exclusive of end index
```

```
'pp'
```

Q: What if you want to take out a substring?

Substring: shorter string from a longer string.

Syntax: x[start index : end index *: step*] # exclusive of end index

Returns the string from “start index” to “end index - 1”.

X[0:2] # slicing:- start index: 0, end index: 2

What is the output of following slicing based code:

```
x = "jack smith"
print(x[2])
print(len(x))
print(x[0:2])
print(x[5:8])
print(x[5:100])
# Python is intelligent about end index.
# If you give an end index larger than the last index, it automatically picks up the last index.
```

Solution:

```
x = "jack smith"
print(x[2])
print(len(x))
print(x[0:2])
print(x[5:8])
print(x[5:100])
```

```
c
10
ja
smi
smith
```

j	a	c	k		s	m	i	t	h
0	1	2	3	4	5	6	7	8	9

Problem on index and slicing

Q: Print all the characters from even indices (inc. 0) of your name.

Q: Print all the characters from odd indices of your name.

Q: Print every second character of your name.

Solution on index and slicing

```
>>> x = 'vikash gupta'
>>> x[0:len(x):2]
'vks ut'
>>>
>>> x[1:len(x):2]
'iahgpa'
>>>
```

```
name1 = "Ashish Jain" # Ahs an
name2 = "Sonia and Johanth"
temp = name1[::2] # Start index, end index, step (default value of step is 1)
print(temp)
temp3 = name2[1::2]
print(temp3)
```

Follow-up Question

Q: What is the difference when single colon is used and when double colon is used?

If there is only one colon while indexing a string:

You have to assume that step is not mentioned and it is equal to 1 (the default value).

name[0::1] # end index is maximum value possible

name[::2] # start index: 0, end index: max value, step: 2

name[:] # start index: 0, end index: len(), step: 1

```
>>> x
'vikash gupta'

>>> x[:]
'vikash gupta'

>>> x[::]
'vikash gupta'
```

Second Set of Questions on Strings

1. Reverse the string

2. Check if a string is a palindrome.

Note: Palindrome is a string that is spelled the same way forward and backward. For example: mam, madam, malayalam.

Solutions

```
>>> x
'vikash gupta'
>>> x[::-1]
'atpug hsakiv'
>>> y = 'madam'
>>> y == y[::-1]
True
>>> z = 'malayalam'
>>> z == z[::-1]
True
>>>
>>> x == x[::-1]
False
```