Overview

1. Assignment the value to variables

2. Types of variables

3. Mathematic operators

4. Basic input and output

5. Comments

Data Types

Integer

- Whole numbers (4, 1000, -400, 10)
- Binary(0b10), octal(0o10), Hexadecimal(0x10)

Float

- Numbers with decimal points, 1.15, 0.4

String

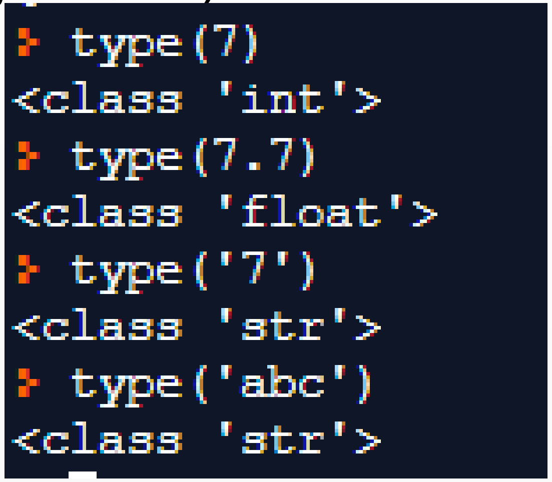
- A text. For example: “word”, “17”, ,“hello world”,
- Blank is also string (it has length)
- The so-called *empty string*, "", has no characters (its length is zero).

Boolean - Truth values (True and False).

List

Tuple

dict



```
> type(7)
<class 'int'>
> type(7.7)
<class 'float'>
> type('7')
<class 'str'>
> type('abc')
<class 'str'>
```

Be careful: 17 is a number, while ‘17’ is a string!

5, -5, 5.8, 10.2, ‘54’, ‘world’, ‘-485.0’

Operators

Assume variable **a** holds the value 10 and variable **b** holds the value 20, then

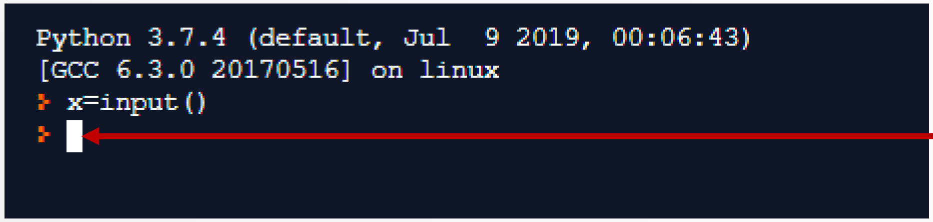
Operator		Description	Example	Shortcut
+	Addition	Adds values on either side of the operator .	a + b = 30	x += y
-	Subtraction	Subtracts right hand operand from left ha nd operand.	a – b = -10	x -= y
*	Multiplication	Multiplies values on either side of the ope rator	a * b = 200	x *= y
/	Division	Divides left hand operand by right hand o perand	b / a = 2.0	x /= y
%	Modulus	Divides left hand operand by right hand o perand and returns **remainder**	b % a = 1	x %= y
**	Exponentiation	Performs exponential (power) calculation on operators	a**b =10 to the power 20	x **= y
//	Integer division	Floor Division - The division of operands where the result is the quotient in which t he digits after the decimal point are remo ved.	9//2 = 4 and 9.0 //2.0 = 4.0, -11// 3 = -4, -11.0//3 = -4.0	x //= y

https://www.tutorialspoint.com/python3/arithmetic_operators_example.htm

Input

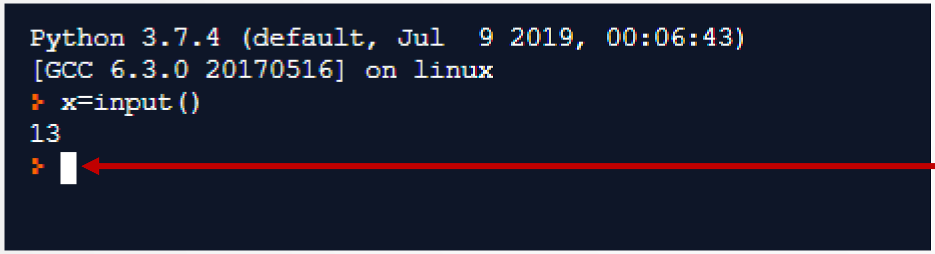
- How do we **input** some data from the user?
- Not surprisingly, using the function input()

```
x = input("Enter any value")
print("The value of x is", x)
```



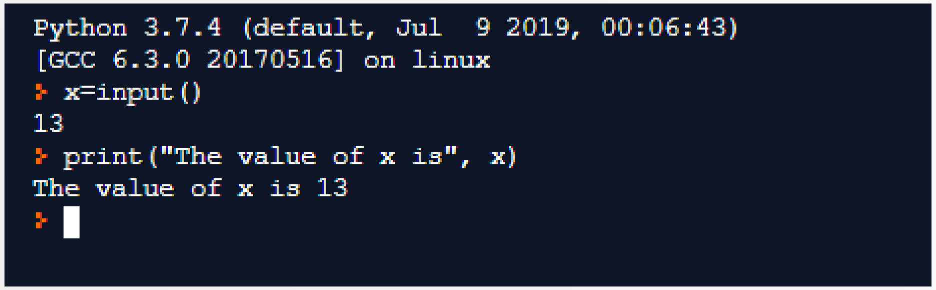
```
Python 3.7.4 (default, Jul  9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
› x=input()
› 
```

The system is waiting your input



```
Python 3.7.4 (default, Jul  9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
› x=input()
13
› 
```

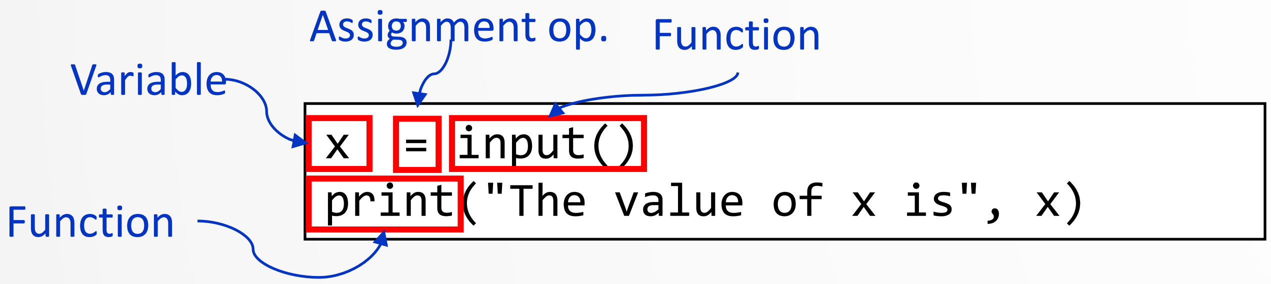
Now the variable x has the input value (i.e., x=13)



```
Python 3.7.4 (default, Jul  9 2019, 00:06:43)
[GCC 6.3.0 20170516] on linux
› x=input()
13
› print("The value of x is", x)
The value of x is 13
› 
```

NAZARBAYEV UNIVERSITY

Input



```
x = input()
print("The value of x is", x)
```

- The function input() reads a sequence of characters from the standard input (usually the user's keyboard) and returns it as a **string**.
- That value is then assigned to the variable x (on the left-hand side of the assignment operator =).
- Whatever is on the right-hand side of the assignment = gets computed first. Then the result is assigned to the variable on the left-hand side. When this is done, the next line of code is executed.
- The function print() now outputs its arguments to the standard output (usually the user's screen), in order in which they were given, separated by a single space character. So,
  - First, a string `"The value of x is"` is written out.
  - Then a singe space character is written out.
  - Then the value of `x` is written out (**not** the string "x" itself, because `x` is a variable!).

Variable - Naming rule

- Rule 1. Name must be comprised of digits(0-9), upper case letters(A-Z), lower case letters(a-z), and the underscore character "_"
- Rule 2. Must begin with a letter or underscore (not digits)
- Rule 3. There are some reserved words which you cannot use as a **variable name** because **Python** uses them for other things

and	assert	in
del	else	raise
from	if	continue
not	pass	finally
while	yield	is
as	break	return
elif	except	def
global	import	for
or	print	lambda
with	class	try
exec		

A, a1, _a, _1a, a_b_c__95

~~1, 1a,~~ 1_

Do not use those keywords as a name of variable

NAZARBAYEV UNIVERSITY

Case of errors

```
>>> and = 3
SyntaxError: invalid syntax
>>> _and = 3
3
>>> 3a = 3
SyntaxError: invalid syntax
>>> aeifh3_3775_38hte = 3
3
>>> aeifh3_3775_38!hte = 3
SyntaxError: invalid syntax
>>> a = 13
>>> print = 3
>>> print(a)
TypeError: 'int' object is not callable
```

Rule 3

Rule 2

Rule 1

Naming rule

Naming variables

- A good name for a variable is short but suggestive of its role: **Circle_Area**
- Can be any (reasonable) length.
- Intuitively understandable

```
>>> Radius = 10
>>> Circle_area = 3.14* Radius* Radius
```

```
>>> R = 10
>>> C_A = 3.14* R* R
```

```
>>> a1 = 10
>>> a2 = 3.14* a1* a1
```

NAZARBAYEV UNIVERSITY

Order of codes

- Order of the code script is important

```
>>> r = 10
>>> A = 3.14 * r * r
>>> print(A)
314.0
```

```
>>> A = 3.14 * r * r
>>> r = 10
>>> print(A)
Traceback (most recent call last):
  File "<pyshell#101>", line 1, in <module>
    A = 3.14*r*r
NameError: name ‘r' is not defined
```

Type conversion: string -> integer

```
>>> a = input(‘Enter any number’)
```

Type conversion: string -> integer

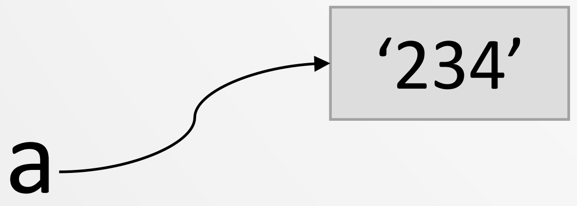
```
>>> a = input(‘Enter any number’)
 ..
```

Type conversion: string -> integer

```
>>> a = input(‘Enter any number’)
‘234’
```

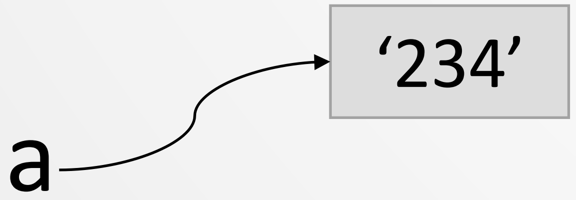
Type conversion: string -> integer

```
>>> a = input('Enter any number')
'234'
```



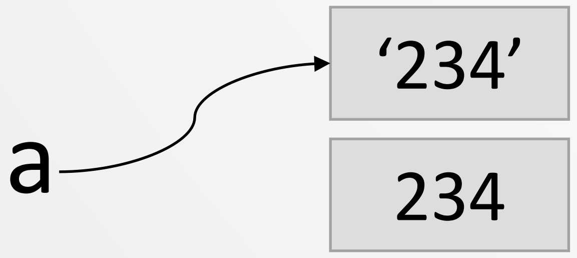
Type conversion: string -> integer

```
>>> a = input('Enter any number')
'234'
>>> type(a)
<class 'str'>
```



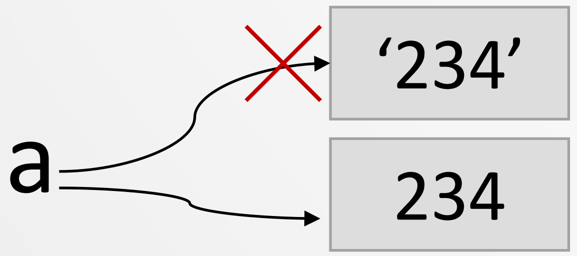
Type conversion: string -> integer

```
>>> a = input(‘Enter any number’)
‘234’
>>> type(a)
<class ‘str’>
>>> int(a)
234
```



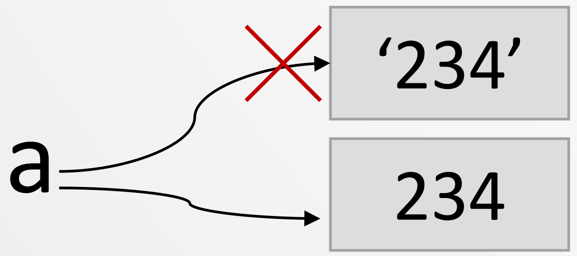
Type conversion: string -> integer

```
>>> a = input(‘Enter any number’)
‘234’
>>> type(a)
<class ‘str’>
>>> int(a)
234
>>> a = int(a)
```



Type conversion: string -> integer

```
>>> a = input('Enter any number')
'234'
>>> type(a)
<class 'str'>
>>> int(a)
234
>>> a = int(a)
>>> type(a)
<class 'int'>
```



Type conversion: float -> string

- Shows how to get a string encoding of a float value.

```
>>> a = -123.45
>>> s = str(a)
>>> print(s)
-123.45
>>> type(s)
<class ‘str’>
```

`str()`- which takes a number (among other allowed types) and converts it to a string.

Type conversion: string * 2

- String values are concatenated by multiplication of integer number

```
>>> a = ‘-123.45’
>>> s = 2 * a
>>> print(s)
‘-123.45-123.45’
>>> type(s)
<class ‘str’>
```

- A string that encodes an integer value can be represented as an int.

```
>>> a = ‘-123’
>>> s = 2 * int(a)
>>> print(s)
-246
```

Type conversion: warning

- int() - which takes a string and converts it to an integer. If the argument is not a string representation of an integer, an error occurs

```
>>> int(‘abc’)
ValueError

>>> float(‘abc’)
ValueError
```

- Converting a floating-point number to an integer is resulted in loosing the decimal points

```
>>> a = -123.45
>>> s = int(a)
>>> print(s)
-123
```

Try to estimate their output

```
>>> str = '-12.45'
>>> str = 2 * str
>>> print(s)
……
>>> type(s)
……
```

```
>>> a = '-49'
>>> s = 2 * int(a)
>>> print(s)
……
```

Try to estimate their output

```
>>> str = ‘-12.45’
>>> str = 2 * str
>>> print(s)
‘-12.45-12.45’
>>> type(s)
……
```

```
>>> a = ‘-49’
>>> s = 2 * int(a)
>>> print(s)
……
```

NAZARBAYEV
UNIVERSITY

Try to estimate their output

```
>>> str = ‘-12.45’
>>> str = 2 * str
>>> print(s)
‘-12.45-12.45’
>>> type(s)
<class ‘str’>
```

```
>>> a = ‘-49’
>>> s = 2 * int(a)
>>> print(s)
……
```

Try to estimate their output

```
>>> str = '-12.45'
>>> str = 2 * str
>>> print(s)
'-12.45-12.45'
>>> type(s)
<class 'str'>
```

```
>>> a = '-49'
>>> s = 2 * int(a)
>>> print(s)
-98
```

Automatic type conversion

- An operation between a **float** and an **int** results in a **float**. So **x** is a **float**.
- Thus, **y** is also a **float** even though its value happens to be an integer.

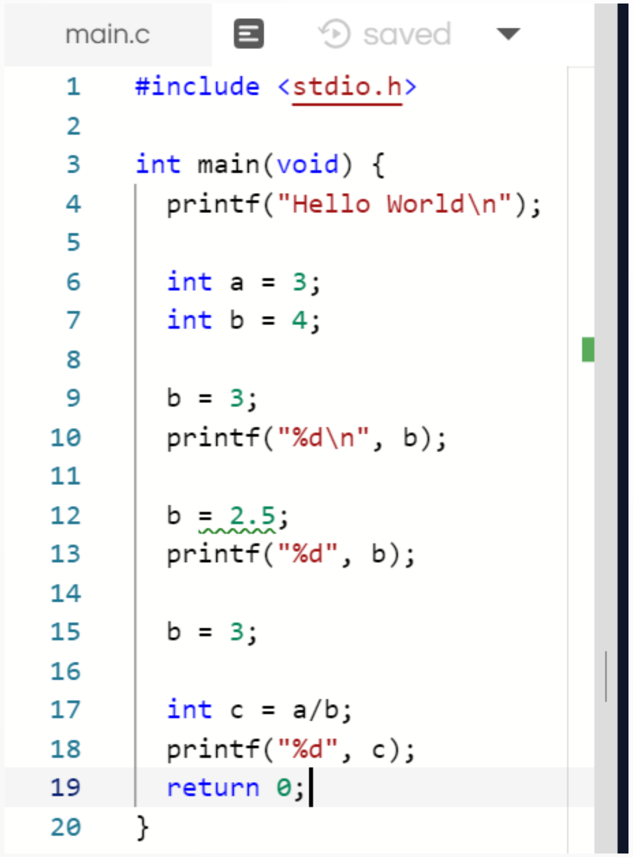
```
>>> x = 1
>>> type(x)
<class ‘int’>
>>> x = x/2
>>> print(x)
0.5
>>> type(x)
<class ‘float’>
>>> y = x*2
>>> print(y)
1.0
>>> type(y)
<class ‘float’>
```

Automatic type conversion

Automatic type conversion

- A variable can hold different types of values at different times.
- In some other languages, the type of a variable is fixed at first definition.

```
>>> x = 'abcde'
>>> type(x)
<class 'str'>
>>> x = 1.0
>>> type(x)
<class 'float'>
>>> x = 32
>>> type(x)
<class 'int'>
```

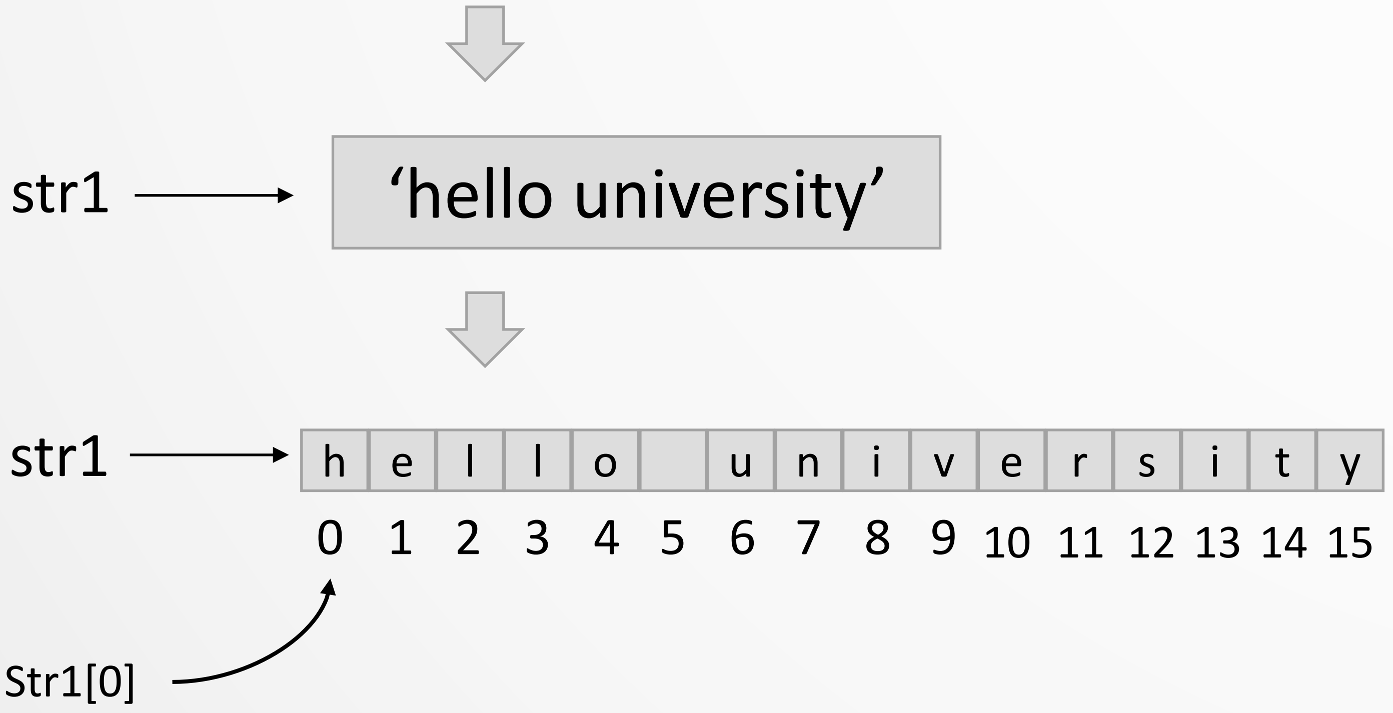


```c
#include <stdio.h>

int main(void) {
    printf("Hello World\n");

    int a = 3;
    int b = 4;

    b = 3;
    printf("%d\n", b);

    b = 2.5;
    printf("%d", b);

    b = 3;

    int c = a/b;
    printf("%d", c);
    return 0;
}
```

Indexing of String-type variable

- The characters in a string can be referenced through their **indices**, called “subscripting”.
- We can access part of a string by index number

```
>>> str1 = ‘hello university’
```



String - index

```
>>> str1 = 'hello university'
```

h	e	l	l	o		u	n	i	v	e	r	s	i	t	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str1[0]
'h'
>>> str1[1]
'e'
>>> str1[-1]
'y'
>>> str1[16]
IndexError: string index out of range
>>> len(str1)
16
```

Indexing over the bounds

Slice with [start:end:step]

- You define a slice by using square brackets, a *start offset*, an *end offset*, and an optional *step size*.

[:] - extracts the entire sequence from start to end.

[start :] - specifies from the start offset to the end.

[: end] - specifies from the beginning to the end offset minus 1.

[start : end] - indicates from the start offset to the end offset minus 1.

[start : end : step] - extracts from the start offset to the end offset minus 1, skipping characters by step.

String - index

```
>>> str1 = ‘hello university’
```

h	e	l	l	o		u	n	i	v	e	r	s	i	t	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str1[3:8]
‘lo un’
>>> str1[3:]
‘lo university’
>>> str1[:3]

>>> str1[6:] + str1[:6]
‘universityhello  ’
>>> str1[6:] + ‘  ‘ + str1[:5]
```

String slicing

Concatenation of each string

NAZARBAYEV UNIVERSITY

String - index

```
>>> str1 = ‘hello university’
```

h	e	l	l	o		u	n	i	v	e	r	s	i	t	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str1[3:8]
‘lo un’
>>> str1[3:]
‘lo university’
>>> str1[:3]
‘hel’
>>> str1[6:] + str1[:6]
‘universityhello ’
>>> str1[6:] + ‘ ’ + str1[:5]
```

String slicing

Concatenation of each string

String - index

```
>>> str1 = ‘hello university’
```

h	e	l	l	o		u	n	i	v	e	r	s	i	t	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str1[3:8]
‘lo un’
>>> str1[3:]
‘lo university’
>>> str1[:3]
‘hel’
>>> str1[6:] + str1[:6]
‘universityhello  ’
>>> str1[6:] + ‘  ‘ + str1[:5]
‘university hello’
```

String slicing

Concatenation of each string

String - index

```
>>> str1 = 'hello university'
```

h	e	l	l	o		u	n	i	v	e	r	s	i	t	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str1[0:10:2]
'hloui'
>>> str1[1: :2]
'el nvriy'
>>> str1[::1]
'hello university'
>>> str1[::-1]
'ytisrevinu olleh'
>>> str1[20:21]
''

>>> str1[-30:-31]
''
```

From offset *0* to *10-1* by step size of *2*, [0 2 4 6 8]

From offset *1* to *end* by step size of *2*, [2 4 6 8 10 12 14 16]

From offset *end* to *end* by step size of *-1*, [-16 -15 … -1]

Given a negative step size, this handy Python slicer can also step backward

NAZARBAYEV UNIVERSITY

String – len()

- Get length of the string with *len()* function

```
>>> str1 = 'Nazarbayev University'
>>> len(str1)
21
>>> empty = ''
>>> len(empty)
0
>>> empty2 = '       '
>>> len(empty2)
7
```

NAZARBAYEV UNIVERSITY

Other functions

- lower, split, replace, etc

```
>>> print('HELLO WORLD'.lower())
hello world
>>> print('HELLO WORLD’.split())
['HELLO', 'WORLD’]
>>> print('HELLO WORLD’.split(‘O’))
['HELL', ' W', 'RLD']
>>> print('HELLO WORLD’.replace(‘HELLO’, ‘MY’))
MY WORLD
```

NAZARBAYEV UNIVERSITY

Boolean returns

Method	True if
str.isalnum()	String consists of only alphanumeric characters (no symbols)
str.isalpha()	String consists of only alphabetic characters (no symbols)
str.islower()	String’s alphabetic characters are all lower case
str.isnumeric()	String consists of only numeric characters
str.isspace()	String consists of only whitespace characters
str.istitle()	String is in title case
str.isupper()	String’s alphabetic characters are all upper case