

Лекция 12

Перевод и адаптация материалов Джона ДеНиро (John DeNero). Используется с разрешения автора.

Объекты

(Пример)

Объекты

Объект — это представление информации.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ
- Особый синтаксис может улучшить структуру программы

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ
- Особый синтаксис может улучшить структуру программы

В Python'е каждое значение — это объект.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ
- Особый синтаксис может улучшить структуру программы

В Python'е каждое значение — это объект.

- У всех объектов есть атрибуты.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ
- Особый синтаксис может улучшить структуру программы

В Python'е каждое значение — это объект.

- У всех объектов есть атрибуты.
- Большая часть обработки данных происходит в методах объекта.

Объекты

Объект — это представление информации.

Он состоит из данных и алгоритмов, объединенных для создания абстракций.

Объекты могут представлять вещи, свойства, взаимодействия и процессы.

Тип объекта называют классом; в Python'е они рассматриваются как значения.

Объектно-ориентированное программирование:

- Метафора для организации больших программ
- Особый синтаксис может улучшить структуру программы

В Python'е каждое значение — это объект.

- У всех объектов есть атрибуты.
- Большая часть обработки данных происходит в методах объекта.
- Функции делают одну вещь; объекты делают множество связанных вещей.

Пример: Строки

(Пример)

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

ASCII Code Chart																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

8 строк: 3 бита

ASCII Code Chart																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

8 строк: 3 бита

16 столбцов: 4 бита

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

8 строк: 3 бита

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа
- Строки 2–5 удобны для 6–битного подмножества (64 элемента)

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

8 строк: 3 бита

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа
- Строки 2–5 удобны для 6–битного подмножества (64 элемента)
- Управляющие символы были предусмотрены для передачи данных

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

«Конец строки» (\n)

ASCII Code Chart																
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

8 строк: 3 бита

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа
- Строки 2–5 удобны для 6-битного подмножества (64 элемента)
- Управляющие символы были предусмотрены для передачи данных

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

«Звонок» (\a)

ASCII Code Chart

«Конец строки» (\n)

8 строк: 3 бита

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	—
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа
- Строки 2–5 удобны для 6–битного подмножества (64 элемента)
- Управляющие символы были предусмотрены для передачи данных

Кодирование строк: стандарт ASCII

American Standard Code for Information Interchange

«Звонок» (\a)

ASCII Code Chart

«Конец строки» (\n)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

8 строк: 3 бита

16 столбцов: 4 бита

- Размещение было выбрано для поддержки сортировки по коду символа
- Строки 2–5 удобны для 6–битного подмножества (64 элемента)
- Управляющие символы были предусмотрены для передачи данных

(Пример)

Кодирование строк: стандарт Unicode

Кодирование строк: стандарт Unicode

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葶	葶

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	苙	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	苙	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聶	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	苙	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
萇	萇	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菴	葳	葳	葵	葶	葶	葶

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE

Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
菖	菖	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菖	葦	葦	葵	葶	葶	葶

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE



Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
菖	菖	荳	扶	葱	吟	荷	葶
8371	8372	8373	8374	8375	8376	8377	8378
葱	菖	葳	葳	葵	葶	葶	葶

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE



Кодирование строк: стандарт Unicode

- 109,000 знаков
- 93 языка (упорядочены)
- Набор свойств знака (например, заглавность или строчность)
- Поддержка двунаправленного порядка вывода
- Каноническое имя для каждого знака

聾	聾	聾	聽	聵	聵	職	瞻
8071	8072	8073	8074	8075	8076	8077	8078
健	腭	腳	腴	暇	股	膈	腸
8171	8172	8173	8174	8175	8176	8177	8178
艱	色	艷	艷	艷	艷	艷	艸
8271	8272	8273	8274	8275	8276	8277	8278
菖	菖	荳	扶	葱	苙	荷	苧
8371	8372	8373	8374	8375	8376	8377	8378
葱	菖	葳	葳	葵	苧	葷	蔥

http://ian-albert.com/unicode_chart/unichart-chinese.jpg

U+0058 LATIN CAPITAL LETTER X

U+263a WHITE SMILING FACE

U+2639 WHITE FROWNING FACE



(Пример)

Кодирование строк: кодировка UTF-8

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты

целые

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

00000000

0

байты

целые

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты

00000000

0

00000001

1

целые

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты	00000000	0	целые
	00000001	1	
	00000010	2	

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты	00000000	0	целые
	00000001	1	
	00000010	2	
	00000011	3	

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты	00000000	0	целые
	00000001	1	
	00000010	2	
	00000011	3	

Код переменной длины: представление чисел различается количеством битов необходимых для их кодирования.

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты	00000000	0	целые
	00000001	1	
	00000010	2	
	00000011	3	

Код переменной длины: представление чисел различается количеством битов необходимых для их кодирования.

В Python'е: длина **строки** измеряется в знаках, длина **представления** в байтах.

Кодирование строк: кодировка UTF-8

UTF (UCS (Universal Character Set) Transformation Format)

Unicode: Связь между знаками и целыми числами

UTF-8: Связь между этими целыми и байтами

Байт — это 8 бит. Байт может кодировать любое целое в интервале 0–255.

байты	00000000	0	целые
	00000001	1	
	00000010	2	
	00000011	3	

Код переменной длины: представление чисел различается количеством битов необходимых для их кодирования.

В Python'е: длина строки измеряется в знаках, длина представления в байтах.

(Пример)

Изменение состояния объекта

Некоторые объекты могут меняться

[Пример]

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:

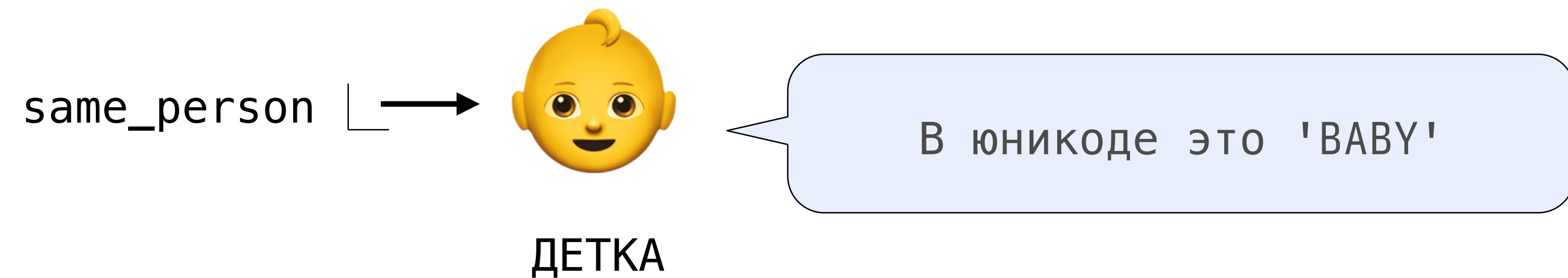
same_person $\mapsto$

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:

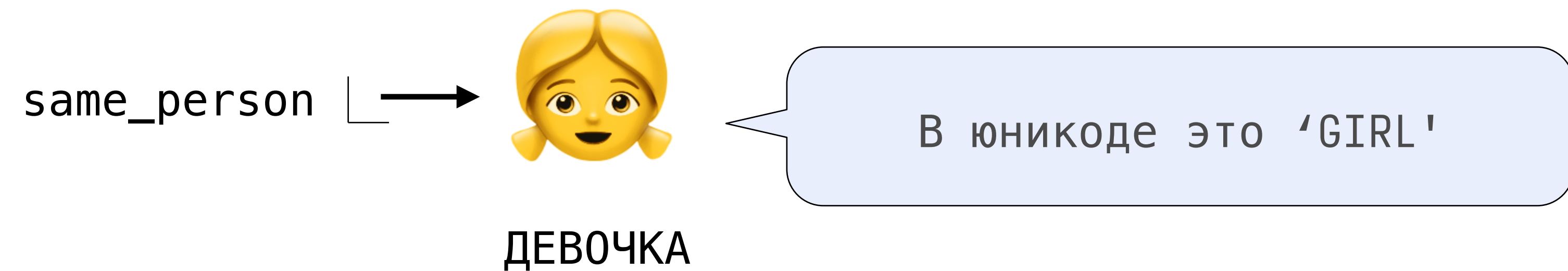


Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:

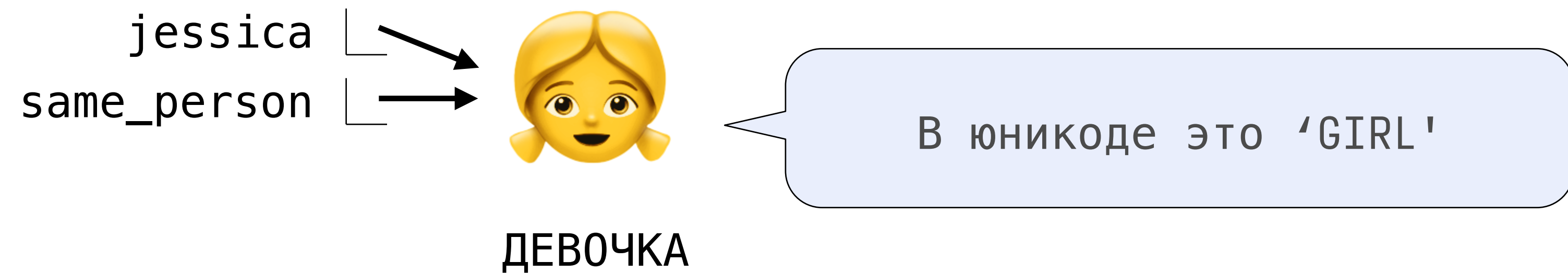


Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:

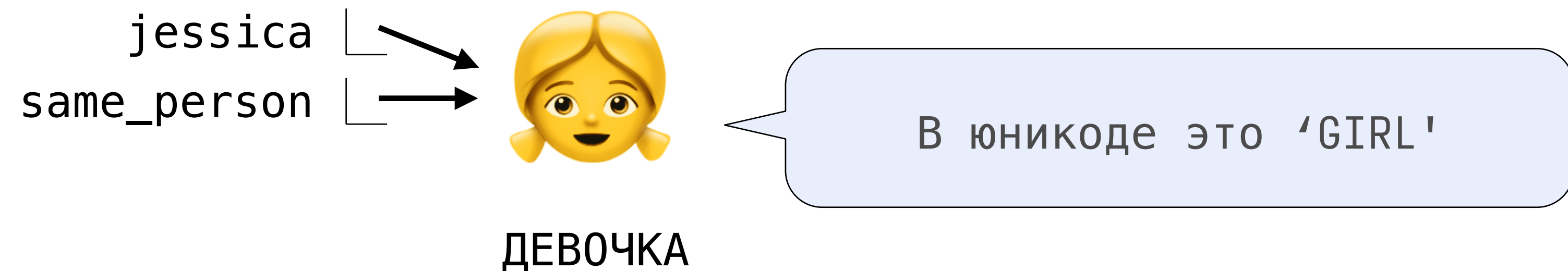


Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:



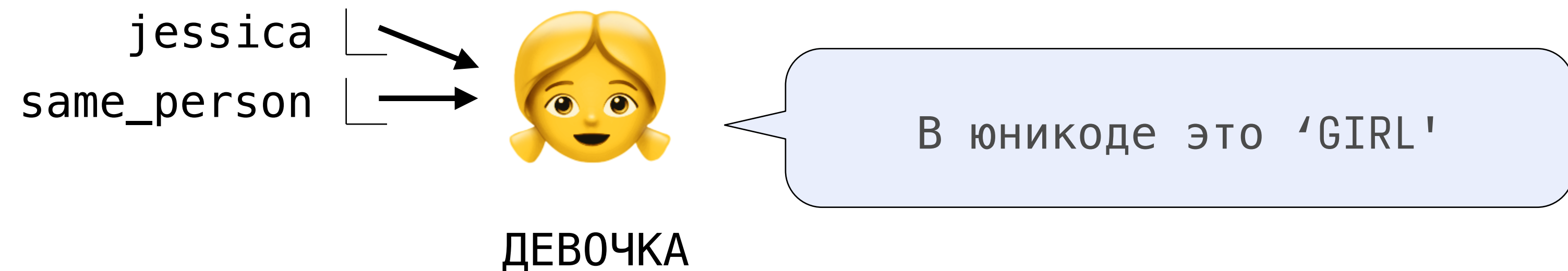
Все имена ссылающиеся на один объект подвержены изменениям.

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:



Все имена ссылающиеся на один объект подвержены изменениям.

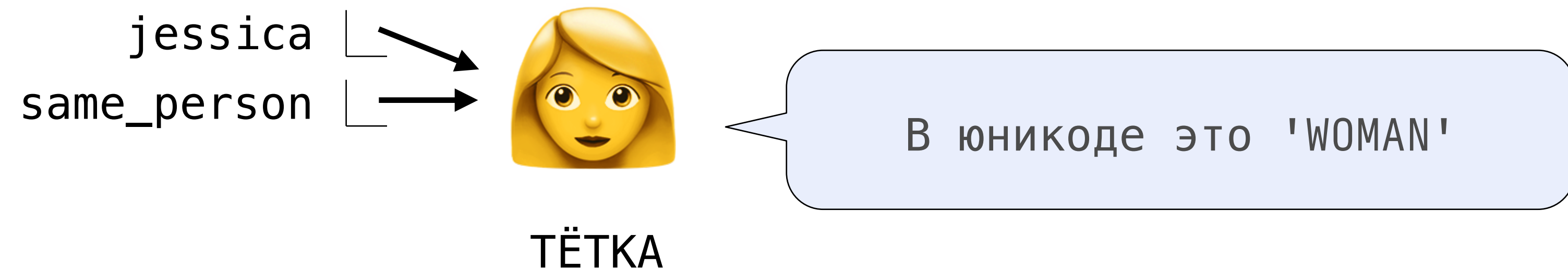
Объекты только *изменяемых* типов могут меняться: списки и словари.

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:



Все имена ссылающиеся на один объект подвержены изменениям.

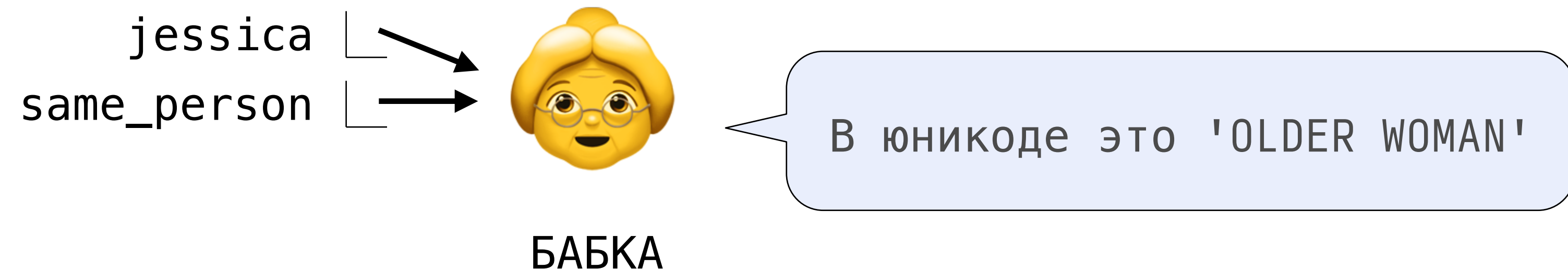
Объекты только *изменяемых* типов могут меняться: списки и словари.

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:



Все имена ссылающиеся на один объект подвержены изменениям.

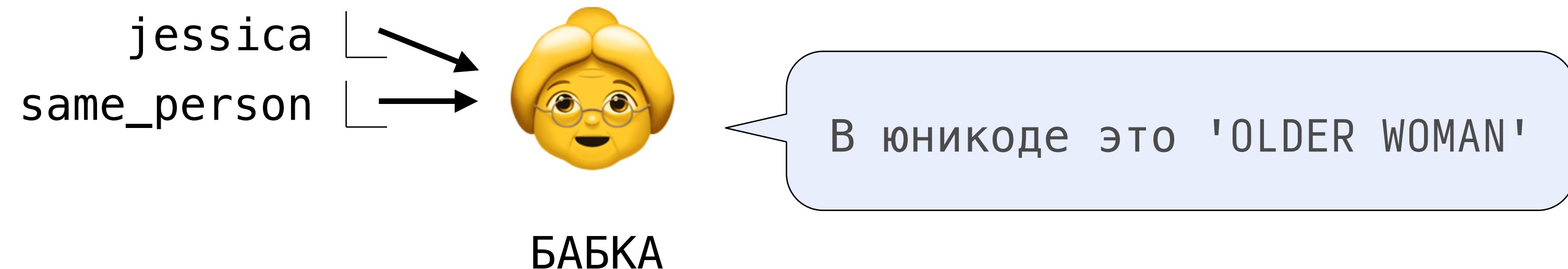
Объекты только *изменяемых* типов могут меняться: списки и словари.

Некоторые объекты могут меняться

[Пример]

Первый пример курса об изменении состояния объекта.

Объект может существенно изменяться в процессе вычисления:



Все имена ссылающиеся на один объект подвержены изменениям.

Объекты только *изменяемых* типов могут меняться: списки и словари.

{Пример}

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
```


Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
def mystery(s):
    s.pop()
    s.pop()
```


Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
def mystery(s):      или def mystery(s):
    s.pop()           s[2:] = []
    s.pop()
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
>>> four = [1, 2, 3, 4]
```

```
def mystery(s):      или def mystery(s):
    s.pop()           s[2:] = []
    s.pop()
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
```

```
def mystery(s):      или def mystery(s):
    s.pop()           s[2:] = []
    s.pop()
```


Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
```

```
>>> len(four)
```

```
4
```

```
>>> mystery(four)
```

```
>>> len(four)
```

```
2
```

```
>>> four = [1, 2, 3, 4]
```

```
>>> len(four)
```

```
4
```

```
>>> another_mystery() # Без аргументов!
```

```
def mystery(s):      или def mystery(s):  
    s.pop()           s[2:] = []  
    s.pop()
```

Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
```

```
>>> len(four)
```

```
4
```

```
>>> mystery(four)
```

```
>>> len(four)
```

```
2
```

```
>>> four = [1, 2, 3, 4]
```

```
>>> len(four)
```

```
4
```

```
>>> another_mystery() # Без аргументов!
```

```
>>> len(four)
```

```
2
```

```
def mystery(s):      или def mystery(s):  
    s.pop()           s[2:] = []  
    s.pop()
```


Изменения могут произойти внутри вызова функции

Функция может изменить любой объект в области видимости.

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
>>> len(four)
2
```

```
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> another_mystery() # Без аргументов!
>>> len(four)
2
```

```
def mystery(s):      или def mystery(s):
    s.pop()
    s.pop()
```

```
def another_mystery():
    four.pop()
    four.pop()
```

Таплы

(Пример)

Таплы — это неизменяемые последовательности

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
```


Туплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

```
>>> turtle = [1, 2, 3]
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

```
>>> turtle = [1, 2, 3]
>>> ooze()
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle  
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

```
>>> x + x
```

Изменение имени:

```
>>> x + x
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

```
>>> x = 2
```

```
>>> x + x
```

Изменение имени:

```
>>> x + x
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

Изменение имени:

```
>>> x + x
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

Изменение имени:

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

```
>>> x + x
```

```
>>> x + x
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

```
>>> x = [1, 2]
```

```
>>> x + x
```

```
>>> x + x
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

```
>>> x = [1, 2]
```

```
>>> x + x
```

```
[1, 2, 1, 2]
```

```
>>> x + x
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

```
>>> x = [1, 2]
```

```
>>> x + x
```

```
[1, 2, 1, 2]
```

```
>>> x.append(3)
```

```
>>> x + x
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
```

```
>>> ooze()
```

```
>>> turtle
```

```
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
```

```
>>> ooze()
```

```
>>> turtle
```

```
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
```

```
>>> x + x
```

```
4
```

```
>>> x = 3
```

```
>>> x + x
```

```
6
```

Изменение объекта:

```
>>> x = [1, 2]
```

```
>>> x + x
```

```
[1, 2, 1, 2]
```

```
>>> x.append(3)
```

```
>>> x + x
```

```
[1, 2, 3, 1, 2, 3]
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ОШИБКА
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ОШИБКА
```

```
>>> s = ([1, 2], 3)
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ОШИБКА
```

```
>>> s = ([1, 2], 3)
>>> s[0][0] = 4
```


Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ОШИБКА
```

```
>>> s = ([1, 2], 3)
>>> s[0][0] = 4
>>> s
```

Таплы — это неизменяемые последовательности

Неизменяемые значения не могут изменяться!

```
>>> turtle = (1, 2, 3)
>>> ooze()
>>> turtle
(1, 2, 3)
```

Следующая лекция: ooze
может изменить связь
имени turtle

```
>>> turtle = [1, 2, 3]
>>> ooze()
>>> turtle
['Здесь может быть что угодно!']
```

Значение выражения может измениться из-за изменений имён или объектов

Изменение имени:

```
>>> x = 2
>>> x + x
4
>>> x = 3
>>> x + x
6
```

Изменение объекта:

```
>>> x = [1, 2]
>>> x + x
[1, 2, 1, 2]
>>> x.append(3)
>>> x + x
[1, 2, 3, 1, 2, 3]
```

Неизменяемая последовательность всё же может измениться если *содержит* изменяемое значение.

```
>>> s = ([1, 2], 3)
>>> s[0] = 4
ОШИБКА
```

```
>>> s = ([1, 2], 3)
>>> s[0][0] = 4
>>> s
([4, 2], 3)
```

Изменение

Одинаковость и изменяемость

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число — это просто числитель и знаменатель.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число — это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число — это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]  
>>> b = a
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
```

Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
>>> b
[10, 20]
```


Одинаковость и изменяемость

- Если отбросить изменяемость объектов, составной объект будет просто объединением своих частей.
- Рациональное число – это просто числитель и знаменатель.
- При учёте изменяемости появляется дополнительное свойство.
- Составной объект приобретает «идентичность» в дополнение к частям, из которых он состоит.
- Список остается «тем же» списком, даже если его содержимое полностью поменялось.
- Противоположно: два списка с одинаковым содержанием, остаются различными.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a.append(20)
>>> a == b
True
>>> a
[10, 20]
>>> b
[10, 20]
```

```
>>> a = [10]
>>> b = [10]
>>> a == b
True
>>> b.append(20)
>>> a
[10]
>>> b
[10, 20]
>>> a == b
False
```


Оператор идентичности

Оператор идентичности

Идентичность

`<expr0> is <expr1>`

вернёт `True` если результаты обоих выражений `<expr0>` и `<expr1>` укажут на один объект

Оператор идентичности

Идентичность

`<expr0> is <expr1>`

вернёт `True` если результаты обоих выражений `<expr0>` и `<expr1>` укажут на один объект

Равенство

`<expr0> == <expr1>`

вернет `True` если результаты обоих выражений `<expr0>` и `<expr1>` будут равными значениями

Идентичные объекты всегда равны

Оператор идентичности

Идентичность

`<expr0> is <expr1>`

вернёт `True` если результаты обоих выражений `<expr0>` и `<expr1>` укажут на один объект

Равенство

`<expr0> == <expr1>`

вернет `True` если результаты обоих выражений `<expr0>` и `<expr1>` будут равными значениями

Идентичные объекты всегда равны

(Пример)

Изменения аргументов по-умолчанию опасны!

Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):  
...     s.append(3)  
...     return len(s)  
...
```

Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):  
...     s.append(3)  
...     return len(s)  
...  
>>> f()  
1
```


Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):  
...     s.append(3)  
...     return len(s)  
...  
>>> f()  
1  
>>> f()  
2
```

Изменения аргументов по-умолчанию опасны!

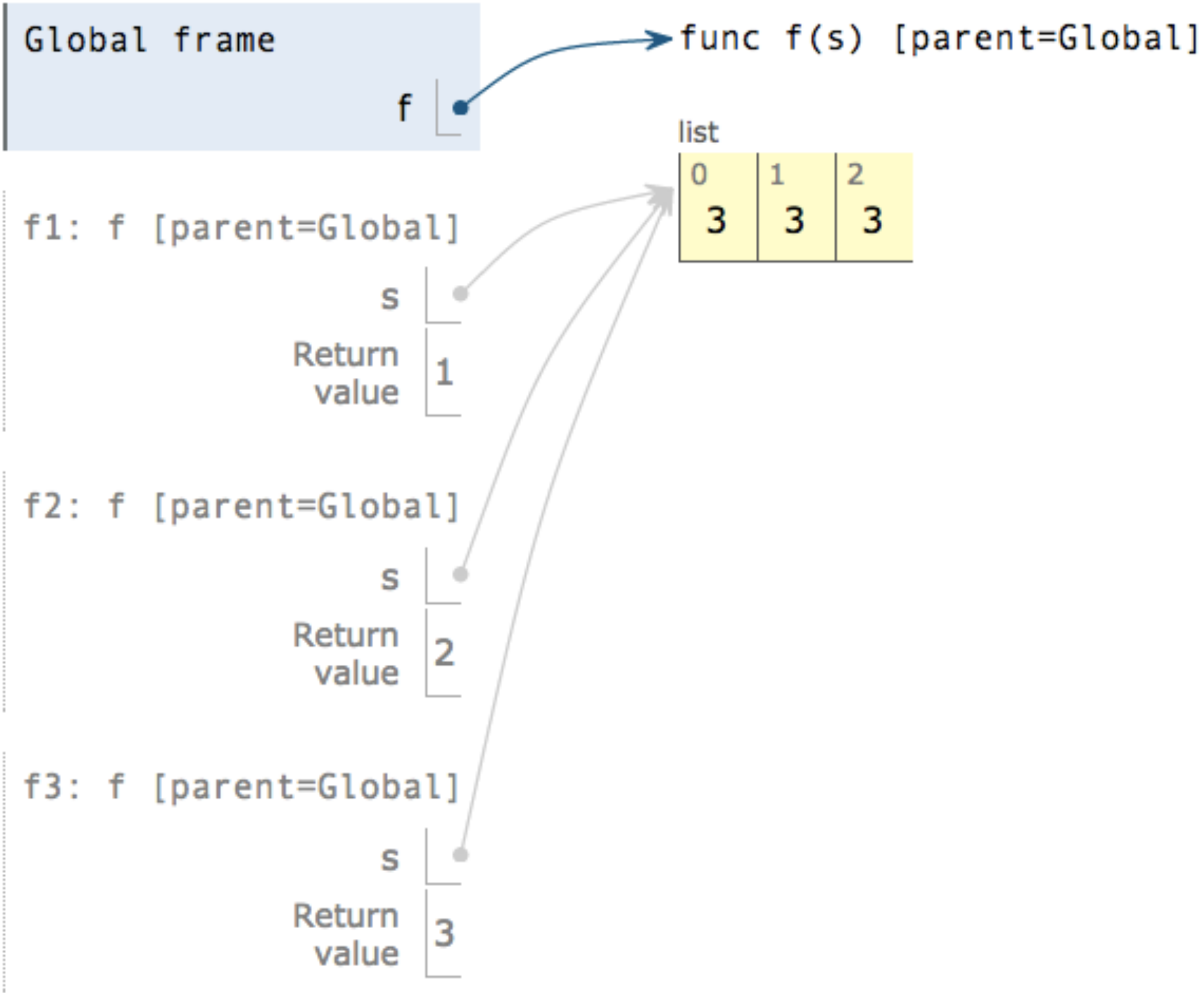
Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):  
...     s.append(3)  
...     return len(s)  
...  
>>> f()  
1  
>>> f()  
2  
>>> f()  
3
```

Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):
...     s.append(3)
...     return len(s)
...
>>> f()
1
>>> f()
2
>>> f()
3
```



Изменения аргументов по-умолчанию опасны!

Значение аргумента по-умолчанию является частью функции и не создается при вызове.

```
>>> def f(s=[]):
...     s.append(3)
...     return len(s)
...
>>> f()
1
>>> f()
2
>>> f()
3
```

