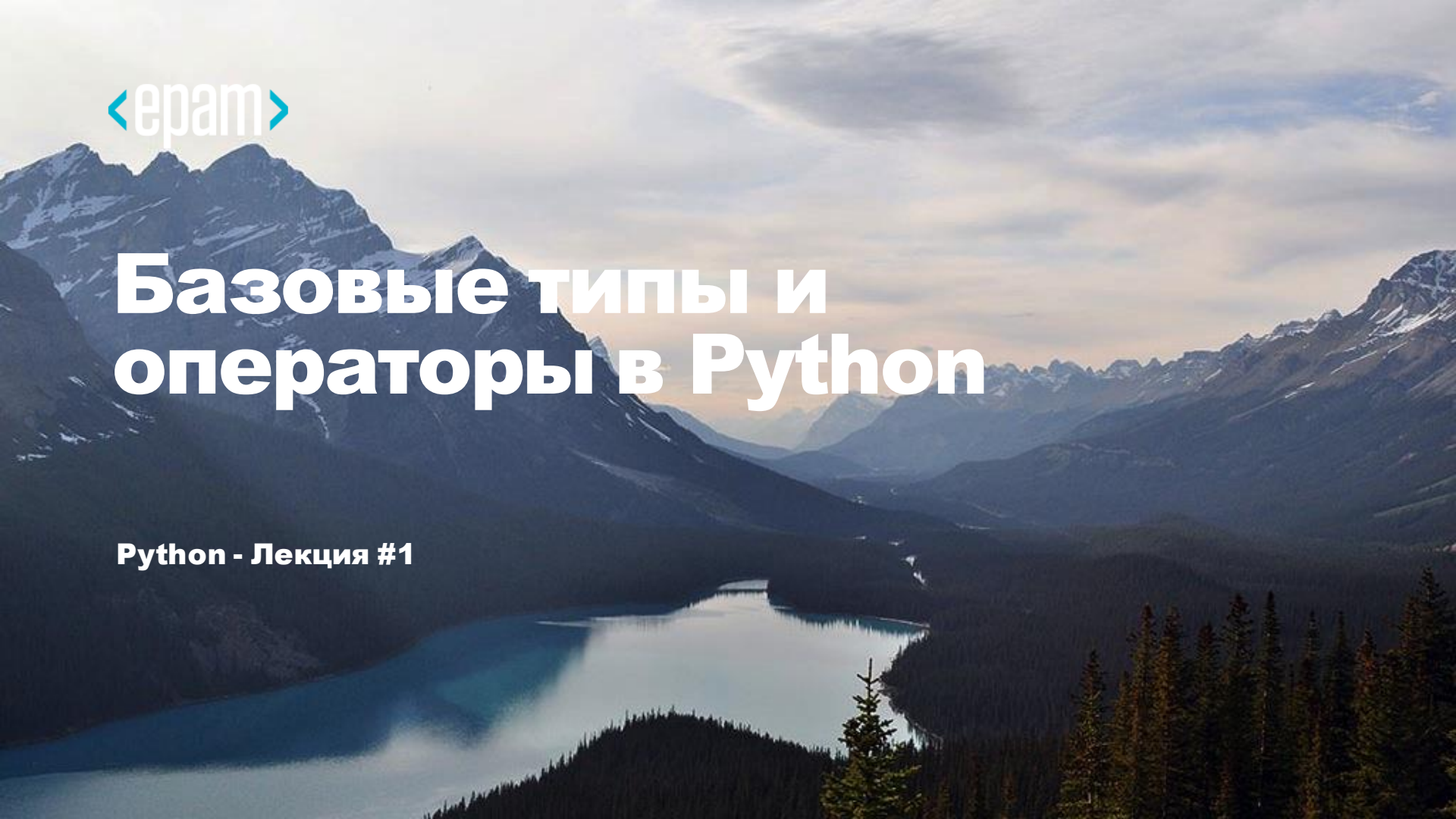


Базовые типы и операторы в Python

Python - Лекция #1

Базовые конструкции

- Регистрозависимый
- _ - для приватных методов
- $ и ? не должны присутствовать в переменных

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Функции для удобства изучения

```
help(obj) -> str
help() -> interactive
help(str) -> str


dir() -> list
dir(str) -> list
dir('123') -> list

locals() -> dict

type(obj) -> type
id(obj)
```

Функции для input / output

```
input(prompt=None) -> str
print(value, sep, end, file, flush) -> None
```

Базовые типы данных

Тип данных	Пример
Numbers	`1234, 3.1415, 3+4j, 0b111, Decimal(), Fraction()`
Strings	`'spam', "Bob's", b'a\x01c', bytes([65, 40]), """multi line"""`
Lists	`[1, [2, 'hello']], list(range(10))`
Dictionaries	`{'food': 'egg', 'taste': 'yum'}, dict(food='egg')`
Tuples	`(1, 'spam')`
Sets	`set('abc')`
Boolean	`True, False`
None	`None`

Числа

Тип	Пример
Integers	`2, 21_100, 0x10, 0o12, 0b1110, (2 ** 10000)`
Floating-point (float)	`3.14, 10., .001, 1e100, 3.14e-10, 0e0, 3.14_15_93`
Complex	`3.14j, 10.j, 10j, .001j, 1e100j, 3.14e-10j, 3.14_15_93j`
Decimals	`decimal.Decimal('10.1002')`
Rationals	`fractions.Fraction(6, 2)`

Строки - форматирование

```
'A has value {a} and b has value {b}'.format(a=a, b=b)
'A has value {} and b has value {}'.format(a, b)
'int: {0:d};  hex: {0:x};  oct: {0:o};  bin: {0:b}'.format(42)
'{:,}'.format(1234567890)
'Correct answers: {:.2%}'.format(19/21)

a = 5
f'a is {a}'  # fastest
```

https://docs.python.org/3.4/library/string.html#formatspec

Type conversion

```
int(str, base=10)
float(str)
str(obj)
```

Task #1

Написать программу, принимающую на вход 2 дроби (вида a/b) (знаменатель одинаковый) и выводящую результат сложения дробей “a/b + c/b = x/b”

Пример: a/b = 1/3 c/b = 5/3
Вывод: 1/3 + 5/3 = 6/3

Task #2

$$? = \frac{12 * x + 25 * b}{1 + x^{2^b}}$$

Bool

- x or y
- x and y
- not x

Всегда False

- None and False.
- 0, 0.0, 0j, Decimal(0), Fraction(0, 1)
- '', (), [], {}, set(), range(0)

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal

Common Sequence Operations #1

Operation	Result
x in s	True if an item of s is equal to x, else False
x not in s	False if an item of s is equal to x, else True
s + t	the concatenation of s and t
s * n or n * s	equivalent to adding s to itself n times

Common Sequence Operations #2

Operation	Result
s[i]	ith item of s, origin 0
s[i:j]	slice of s from i to j
s[i:j:k]	slice of s from i to j with step k

Common Sequence Operations #3

Operation	Result
len(s)	length of s
min(s)	smallest item of s
max(s)	largest item of s
s.index(x[, i[, j]])	index of the first occurrence of x in s (at or after index i and before index j)
s.count(x)	total number of occurrences of x in s

Mutable Sequence Operations #1

Operation	Result
`s[i] = x`	item i of s is replaced by x
`s[i:j] = t`	slice of s from i to j is replaced by the contents of the iterable t
`del s[i:j]`	same as s[i:j] = []
`s[i:j:k] = t`	the elements of s[i:j:k] are replaced by those of t
`del s[i:j:k]`	removes the elements of s[i:j:k] from the list

Mutable Sequence Operations #2

Operation	Result
`s.append(x)`	appends x to the end of the sequence (same as s[len(s):len(s)] = [x])
`s.clear()`	removes all items from s (same as del s[:])
`s.copy()`	creates a shallow copy of s (same as s[:])
`s.extend(t) or s += t`	extends s with the contents of t (for the most part the same as s[len(s):len(s)] =t)
`s *= n`	updates s with its contents repeated n times

List

- []
- [a], [a, b, c]
- [x for x in iterable]
- list() or list(iterable)

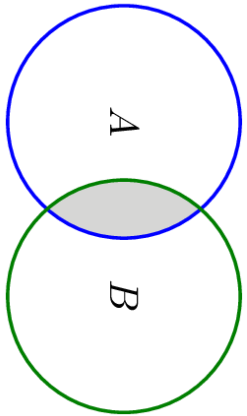
list.sort()

Tuple

- ()
- (a,)
- (a, b, c)
- tuple() or tuple(iterable)

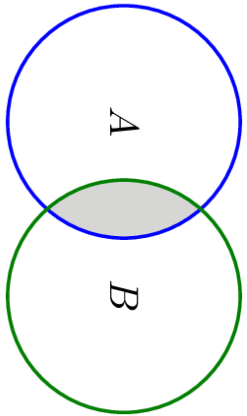
Set

len(s)	Размер
x in s	Проверка присутствия
x not in s	Проверка не присутствия
isdisjoint(other)	Если нет общих элементов
issubset(other), set <= other, set < other	Если является включенным сетом
issuperset(other), set >= other, set > other	Если включает сет
union(*others), set | other | ...	Объединение
intersection(*others), set & other & ...	Пересечение
difference(*others), set - other - ...	Разница
symmetric_difference(other), set ^ other	Элементы без общих



Set - updates

update(*others), set |= other | ...	Добавить
intersection_update(*others), set &= other & ...	Оставить только общие
difference_update(*others), set -= other | ...	Удалить элементы
symmetric_difference_update(other), set ^= other	Оставить только diff
add(elem)	Добавить
remove(elem)	Удалить с ошибкой
discard(elem)	Удалить без ошибки
pop()¶	«Вытащить» элемент
clear()	Очистить
copy()	Сделать копию



Dict

- a = dict(one=1, two=2, three=3)
- b = {'one': 1, 'two': 2, 'three': 3}
- = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
- d = dict([('two', 2), ('one', 1), ('three', 3)])
- e = dict({'three': 3, 'one': 1, 'two': 2})
- a == b == c == d == e

Dictionary view

Результат dict.keys(), dict.values() dict.items() – dictview
Необходимо приводить

If / else

```
if COND then:
  pass
else:
  pass
```

```
if COND1 then:
  pass
elif COND2:
  pass
elif COND3:
  pass
else:
  pass
```

```
x = a if COND else b
```

For

```
for item in ITERABLE:
  pass
```

```
for key in DICT:
  pass
```

```
for key, value in DICT.items():
  pass
```

```
for n, item in enumerate(ITERABLE):
  pass
```

For

```
break
continue
```

For / else

```
x = int(input())

for i in [1, 2, 3]:
  if i == x:
    break
else:
  print('Not found')
```

While

```
while COND:
  pass
```

Task #3 - FizzBuzz

Напишите программу, которая выводит на экран числа от 1 до 100. При этом вместо чисел, кратных трем, программа должна выводить слово Fizz, а вместо чисел, кратных пяти — слово Buzz. Если число кратно пятнадцати, то программа должна выводить слово FizzBuzz.

HW #1

Изучить функции str, int, dict, set, list, tuple, byte, bytearray

dir(str), dir(int), ….

https://docs.python.org/3/library/stdtypes.html