Лекция 2

Циклы.

Списки и строки.

Циклы: while

```
sum_1 = 0
for i in range(11):
    sum_1 += i * i

sum_2 = 0
i = 0
while i <= 10:
    sum_2 += i * i
    i += 1

print(sum_1, sum_2)
```

(цикл for)

(цикл while)

```
385 385
```

Циклы: break

```
number = 23
guess = 0

while True:
    guess = int(input('Try to guess my number: '))
    if number == guess:
        print('Well done!')
        break
    elif number < guess:
        print('My number is less than yours.')
    else:
        print('My number is greater than yours.')
```

```
>Try to guess my number: 20
My number is less than yours.
>Try to guess my number: 23
Well done!
```

Циклы: continue

```
number = 23
guess = 0

while True:
    guess = int(input('Try to guess my number: '))
    if guess > 100 or guess < 0:
        print('It is from 0 to 100, take it easy.')
        continue
elif number == guess:
    print('Well done!')
    break
elif number < guess:
    print('My number is less than yours.')
else:
    print('My number is greater than yours.')
```

Циклы: else

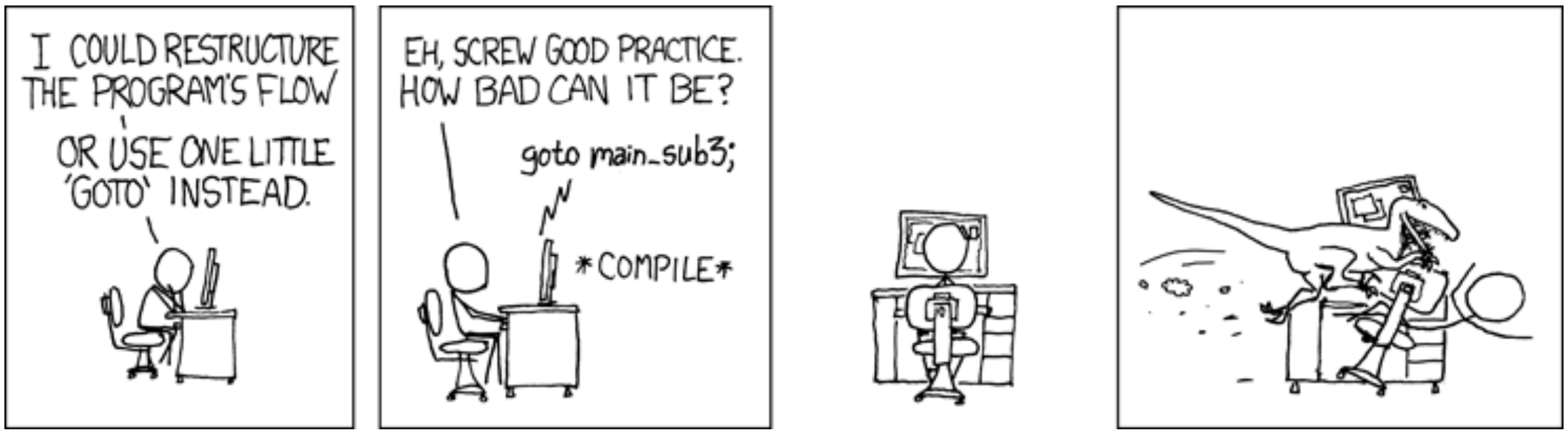
```
number = 23
guess = 0

for attempt in range(1, 8):
    guess = int(input('Try to guess my number:'))
    if number == guess:
        print('Well done!')
        break
    elif number < guess:
        print('My number is less than yours.')
    else:
        print('My number is greater than yours.')
else:
    print('Sorry, you are out of attempts')
```

Циклы: советы

- Можно писать **for** i **in** lst:
где lst - произвольный список (об этом позже)

- Если заранее известно число итераций цикла, предпочтительнее использовать **for**.

- **break** и **continue** бывают полезны и необходимы. Но часто лучше обойтись без них.



Запятая

```
a = 1
b = 2
# ok: save one line
a, b = 1, 2

# bad: decrease readability
name, n, my_sum = input(), int(input()), 0

tmp = a
a = b
b = tmp
# great: save 2 lines & increase readability
a, b = b, a
```

Списки

```
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri',
'Sat', 'Sun']
day_numbers = [1, 2, 3, 4, 5, 6, 7]
weird_list = [42, None, 'kill', ['me']]
print(days[1], day_numbers[1])
```

```
empty = list()
print(empty)

alphabet = list('abcdefghijklmnopqrstuvwxyz')
print(alphabet[13], len(alphabet))
```

```
Tue 2
[]
n 26
```

Списки: срезы

```
a = list(range(6)) # a = [0, 1, 2, 3, 4, 5]
print(a[0], a[-1])
print(a[1:5], a[1:-1])
print(a[2:], a[:2])
print(a[-2:], a[:-2])
print(a[::2], a[2::2]) # compare with range args
print(a[::-1]) # popular trick
```

```
0 5
[1, 2, 3, 4] [1, 2, 3, 4]
[2, 3, 4, 5] [0, 1]
[4, 5] [0, 1, 2, 3]
[0, 2, 4] [2, 4]
[5, 4, 3, 2, 1, 0]
```

Списки: изменение

```
a = ['2', 'bee', 'or', 'what?']
a[0] = 'To'
a.pop()
print(a)
a.remove('bee')
a.insert(1, 'be') # index, value
print(a)
a.append('not')
a.extend(['to', 'be?'])
print(a)
```

```
['To', 'bee', 'or']
['To', 'be', 'or']
['To', 'be', 'or', 'not', 'to', 'be?']
```

Списки: замечания

***Быстрые* операции (работают за O(1)):**

- добавить в конец (*append*)
- убрать с конца (*pop*)
- обратиться/изменить по индексу (*[]*)
- узнать длину (*len*)

***Медленные* операции (работают за O(n)):**

- добавить в произвольное место (*insert*)
- удалить по значению (*remove*)
- проверить принадлежность (*in*)

Медленные методы лучше не использовать даже при работе с небольшими данными.

Про асимптотическое время работы будет рассказано сегодня на факультативе.

Неизменяемые списки (кортежи)

```
a = (1, 2, 3)
b = tuple() # b = ()
c = (3,) # comma is required

l = [0, 1, 2, 6, 4, 5]
t = tuple(l)
print(t[0], t[-2:])
l[3] = 3
t[3] = 3 # error
```

```
0 (4, 5)
TypeError: 'tuple' object doesn't support item assignment
```

Кортежи нужны для словарей - подробнее на следующих лекциях.

Строки

- Можно обращаться, как к массивам:

```
s = 'string'
l = list(s)
print(s[:3], l[:3])
```

- Нельзя изменять.

```
#s[:3] = 'aight'
l[3:] = list('aight')
s = s[:3] + 'aight'
print(s)
```

```
str ['s', 't', 'r']
straight
```

Строки: методы

```
e = '1,2,3'
print(e.find(',')) # first occurrence
f = 'Time flies like a arrow; fruit flies like a banana.'
f = f.replace('a ', 'an `, 1) # first occurrence
print(f)
g = 'goooooogle'
print(g.count('o')) # also works with lists
```

```
1
Time flies like an arrow; fruit flies like an banana.
6
```

Строки: методы

```
h = '  Hello, world!      '
print(h.strip()) # also lstrip() and rstrip()
i = 'i aM 11 yEaRs OLd'
print(i.lower())
k = '345'
print(k.isdigit()) # and so on
```

```
Hello, world!
i am 11 years old
True
```

Split и join

```
sent = 'fall leaves as soon as leaves fall'
words = sent.split()
words = words[::-1]
sent = ' '.join(words)
print(sent)
```

```
nums = '1,2,3,4'
print('\n'.join(nums.split(',')))
```

```
fall leaves as soon as leaves fall
1
2
3
4
```

Чтение текста

```
# Read multiple lines
n = int(input())
lines = []
for i in range(n):
    lines.append(input())
text = ' '.join(lines)
words = text.split()
print(words)
```

```
>3
>An old silent pond…
>A frog jumps into the pond.
>Splash! Silence again.
['An', 'old', 'silent', 'pond...', 'A', 'frog', 'jumps',
'into', 'the', 'pond.', 'Splash!', 'Silence', 'again.']
```

Про чтение списка чисел можно прочесть в pythontutor: списки.

Генераторы списков

```
zeroes = [0, 0, 0, 0, 0]
zeroes = [0] * 5
zeroes = [0 for i in range(5)]
squares = [i * i for i in range(5)]
s = ' '.join([str(i) for i in squares])
print(s)
```

```
ints = [int(i) for i in input().split()]
print(sum(ints))
```

```
0 1 4 9 16
15
```

Подробнее про генераторы и функциональное программирование — на следующих лекциях.

Встроенная документация

- **help()**: показывает краткую информацию о функциях и методах.

  Примеры: `help([].append)`, `help(sum)`

- **dir()**: показывает список методов для данного объекта (переменной или типа). Методы, начинающиеся и оканчивающиеся на __ - специальные: про них будет рассказано позже.

  Примеры: `dir(str)`, `dir(3)`

Полезные ссылки и книги

- https://docs.python.org/3/library/index.html - стандартная библиотека python 3 (eng). Здесь можно более подробно узнать про конкретную функцию или особенности синтаксиса.

- https://wiki.python.org/moin/TimeComplexity - таблица времени исполнения (асимптотического) стандартных операций со списками и другими структурами.

- Swaroop C.H. - A byte of python. - введение в python для не-программистов на русском языке.

- http://informatics.mccme.ru/course/view.php?id=156 - исходный курс Дениса Кириенко, ориентированный для школьников. На его основе был создан pythontutor.ru