wiedzmy

CZĘŚĆ 1 wstęp do nauki pythona

Zakres:

1. Setting up the environment (Anaconda, Spyder)
2. Basic data types, type conversion
3. Mathematical operators
4. Logical operators
5. Defining variables
6. If statements
7. While loops
8. For loops

Setting up the environment

https://www.anaconda.com/distribution/

1. Wchodzimy w powyższy link i pobieramy wersję Python 3.7

2. Windows - otwieramy Anacondę z paska startowego; Mac - Anaconda Navigator.

3. Otwieramy aplikację Spyder.

4. W Ustawieniach preferencji użytkownika wchodzimy w zakładkę Run i zaznaczamy "Clear variables before execution"

BASIC DATA TYPES

NUMBERS

Floats

3.14159
5.6
7.0

```
In: type(3.7) == float
Out: True
```

Integers

5
56
70

```
In: type(3) == int
Out: True
```

STRINGS

“hello”
‘My name is Magda’

```
In: type(“hello world!”) == str
Out: True
```

We can use both “double” and ‘single’ quotes to define a string.

BOOLEAN

True / False

```
In: type(True) == bool
Out: True
```

TYPE CONVERSION

Int -> Float

```
In: x = 3
In: x = float(x)
In: x
Out: 3.0
```

Float -> Int

```
In: x = 5.7345
In: x = int(x)
In: x
Out: 5
```

Notice that float() always rounds down to the closest integer.

Int/Float -> String

```
In: x = 3
In: x = str(x)
In: x
Out: ‘3’
```

Boolean -> Str

```
In: str(False)
Out: ‘False’
```

Expression -> Boolean

```
In: bool(1)
Out: True
In: bool(0)
Out: False

In: bool(type(‘hello’) == str)
Out: True
In: bool(‘hello’)
Out: True
```

MATHEMATICAL OPERATORS

=

ASSIGNMENT

Equal sign in python is used to assign a value to a variable.

```
In: x = 10
```

In this statement we bind the value of 10 to the variable x. Later when we call x, our machine will remember what we’ve stored under that name and return the value.

```
In: x
Out 10
```

+

ADDITION

```
In: x = 10
In: y = 11
In: x + y
Out: 22

In: ‘cześć’ + ‘ ‘ + ‘dziewczyny’
Out: cześć dziewczyny
```

+=

INCREMENT

```
In: x = 10
In: x += 5
In: x
Out: 15
```

expression ‘x += 5’ means: take value of x and add 5 to it.
x += 5 == x = x + 5

SUBSTRACTION

```
In: x = 10
In: y = 11
In: x - y
Out: -1
```

-=

DECREMENT

```
In: x = 10
In: x -= 5
In: x
Out: 5
```

expression ‘x -= 5’ means: take value of x and decrement its value by 5.
x -= 5 == x = x - 5

/

DIVISION

```
In: x = 10
In: y = 2
In: x / y
Out: 5

type(10/2 == float)
```

//

FLOOR DIVISION

```
In: x = 10
In: y = 3
In: x // y
Out: 3
type(10//3) == int

In: x = 10.0
In: y = 3
In: x // y
Out: 3.0
```

*

MULTIPLICATION

```
In: x = 10
In: y = 2
In: x * y
Out: 20
```

**

POWER

```
In: x = 10
In: y = 2
In: x ** y
Out: 100
```

==

EQUALITY

In order to check if two values are equal we use == (since = is already taken)

```
In: x = 10
In: y = 10
In: x == y
Out: True
```

> < >= <=

COMPARASION

```
In: x = 10
In: y = 15
In: x > y
Out: False
In: x < y
Out: True
In: x >= y
Out: False
In: x = 15
In: x <= y
Out: True
```

!=

NOT EQUAL

```
In: x = 10
In: y = 11
In: x != y
Out: True
```

%

MODULUS

remainder of the division of left operand by the right

```
In: x = 10
In: y = 2
In: x % y
Out: 0
In: 13%5
Out: 3
```

INTEGER/FLOAT CONVERSION

integer + integer = integer

```
In: x = 10
In: y = 2
In: x + y
Out: 12
In: type(x + y)
Out: int
```

integer + float = float

```
In: x = 10
In: y = 0.2
In: x + y
Out: 10.2
In: type(x + y)
Out: float
```

float + float = float

```
In: x = 10.8
In: y = 2.2
In: x + y
Out: 13.0
In: type(x + y)
Out: float
```

float - float = float

```
In: x = 10.2
In: y = 0.2
In: x + y
Out: 10
In: type(x + y)
Out: float
```

integer - integer = integer

```
In: x = 10
In: y = 2
In: x - y
Out: 8
In: type(x - y)
Out: int
```

float - integer = float

```
In: x = 10.8
In: y = 2
In: x - y
Out: 8.8
In: type(x + y)
Out: float
```

integer * integer = integer

```
In: x = 10
In: y = 2
In: x * y
Out: 20
In: type(x * y)
Out: int
```

float * integer = float

```
In: x = 10.0
In: y = 2
In: x * y
Out: 20.0
In: type(x * y)
Out: float
```

float * float = float

```
In: x = 10.2
In: y = 2.0
In: x * y
Out: 20.4
In: type(x * y)
Out: float
```

integer / integer = float

```
In: x = 10.0
In: y = 2
In: x / y
Out: 5.0
In: type(x / y)
Out: float
```

integer / float = float

```
In: x = 10
In: y = 2.0
In: x / y
Out: 5.0
In: type(x / y)
Out: float
```

integer // integer = integer

```
In: x = 10
In: y = 2
In: x / y
Out: 5
In: type(x / y)
Out: int
```

integer // float = integer

```
In: x = 10
In: y = 3.0
In: x // y
Out: 3
In: type(x // y)
Out: int
```

float // float = float

```
In: x = 10.5
In: y = 2.0
In: x // y
Out: 5.0
In: type(x // y)
Out: float
```

LOGICAL OPERATORS

wiedzmy

and

```
In: x = 10
In: y = 7
In: x < 15 and y < 10
Out: True

In: 1 and 0
Out: 0

In: 1 and 1
Out: 1

In: False and False
Out: False

In: 1 and True
Out: True
```

or

```
In: x = 10
In: y = 12
In: x < 15 or y < 10
Out: True

In: 1 or 0
Out: 1

In: 0 or 0
Out: 0

In: False or True
Out: False

In: 1 or True
Out: 1
In: True or 1
Out: True
```

DEFINING VARIABLES

wiedzzmy

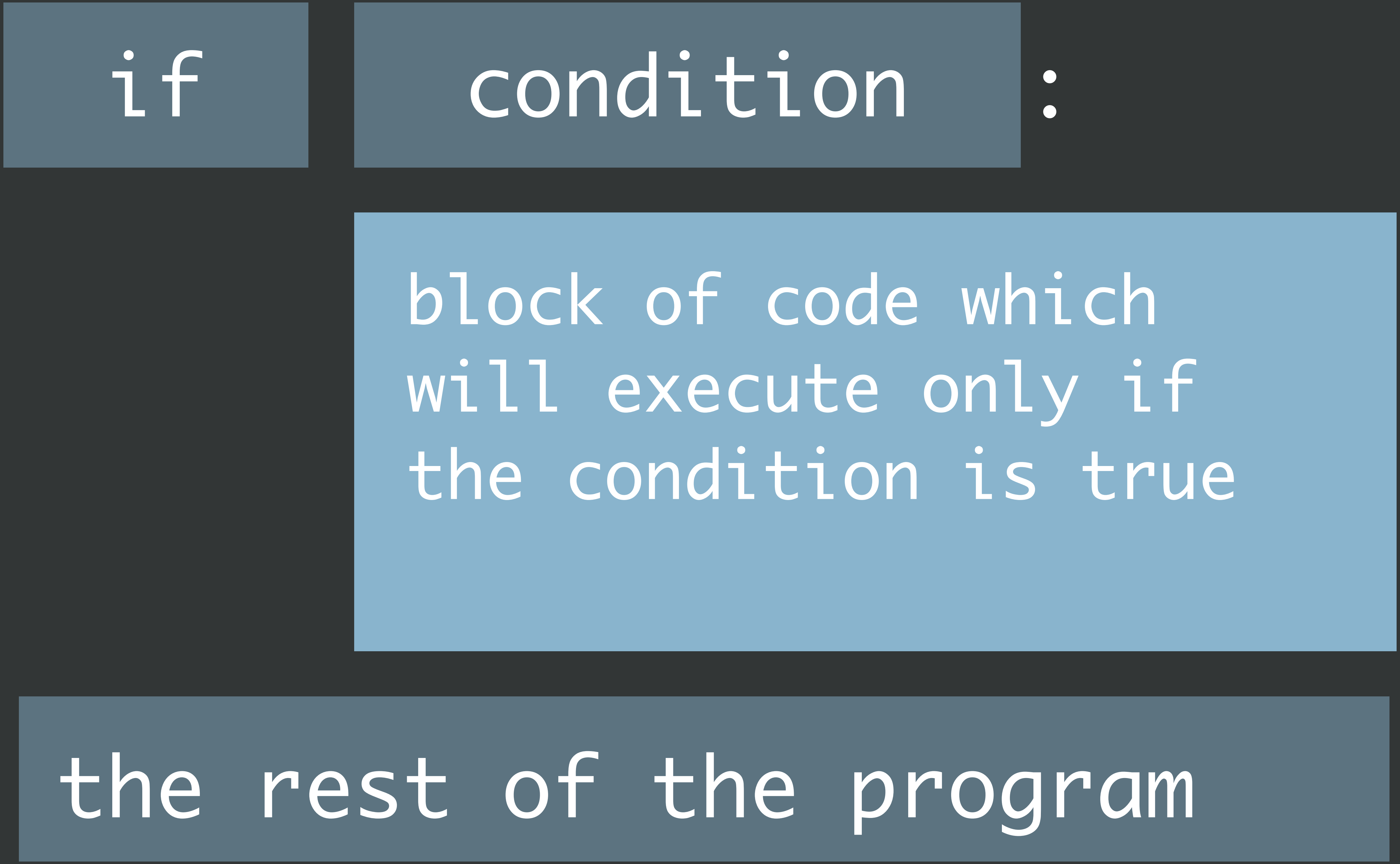
```
In: x = 10
In: x
Out: 10

In: x = “Hello ” + “girls”
In: x
Out: “Hello girls”

In: x = “Hello” + “girls”
In: x
Out: “Hellogirls”

In: x = “Hello”
In: y = “girls”
In: print(x, y)
Out: Hello girls
In: x + y
Out: “Hellogirls”
```

IF STATEMENTS



if condition :

block of code which will execute only if the condition is true

the rest of the program

```
In: x = 10
In: if x == 10:
      print(‘x is equal to 10’)
Out: ‘x is equal to 10’
```

```
In: x = 10
In: if x == 10:
      print(‘x is equal to 10’)
Out: ‘x is equal to 10’
```

Block of code will execute only if condition after ‘if’ keyword is true.

```
In: x = 11
In: if x == 10:
      print(‘x is equal to 10’)
There is no output.
```

Block of code will execute only if condition after ‘if’ keyword is true.

IF / ELIF/ ELSE

```
if condition :
    block of code which will execute only
    if a condition is true
else :
    block of code which will execute only
    if both conditions are false
```

```
In: x = 10
In: if x == 10:
      print(‘x is equal to 10’)
    else:
      print(‘x is not equal to 10’)
Out: ‘x is equal to 10’
```

```
In: x = 10
In: if x == 10:
      print(‘x is equal to 10’)
    else:
      print(‘x is not equal to 10’)
Out: ‘x is equal to 10’
```

Block of code will execute only if condition after ‘if’ keyword is true. In this case code after ‘else’ keyword won’t execute.

```
In: x = 11
In: if x == 10:
      print(‘x is equal to 10’)
    else:
      print(‘x is not equal to 10’)
Out: ‘x is not equal to 10’
```

condition after ‘if’ keyword is false. In that case our program will execute block of code in else statement.

```
if first condition :
    block of code which will execute only
    if the first condition is true
elif next condition :
    block of code which will execute only
    if first condition is false and the
    next condition is true
else :
    block of code which will execute only
    if both conditions are false
```

```
In: x = 10
In: if x == 10:
       print(‘x is equal to 10’)
    elif x == 11:
       print(‘x is equal to 11’)
    elif x == 12:
       print(‘x is equal to 12’)
    else:
       print(‘x is not equal to 10, 11 or 12’)
Out: ‘x is equal to 10’
```

```
In: x = 10
In: if x == 10:
       print(‘x is equal to 10’)
    elif x == 11:
       print(‘x is equal to 11’)
    elif x == 12:
       print(‘x is equal to 12’)
    else:
       print(‘x is not equal to 10, 11 or 12’)
Out: ‘x is equal to 10’
```

The condition after ‘if’ keyword is true. In this case only this block of code will execute, our program will not evaluate elif’s/else.

```
In: x = 11
In: if x == 10:
      print(‘x is equal to 10’)
    elif x == 11:
      print(‘x is equal to 11’)
    elif x == 12:
      print(‘x is equal to 12’)
    else:
      print(‘x is not equal to 10, 11 or 12’)
Out: ‘x is equal to 11’
```

If the condition directly after ‘if’ statement is false, our program will jump to the next condition (there is no upper limit to number of elif’s).

```
In: x = 35
In: if x == 10:
         print(‘x is equal to 10’)
     elif x == 11:
         print(‘x is equal to 11’)
     elif x == 12:
         print(‘x is equal to 12’)
     else:
         print(‘x is not equal to 10, 11 or 12’)
Out: ‘x is not equal to 10, 11 or 12’
```

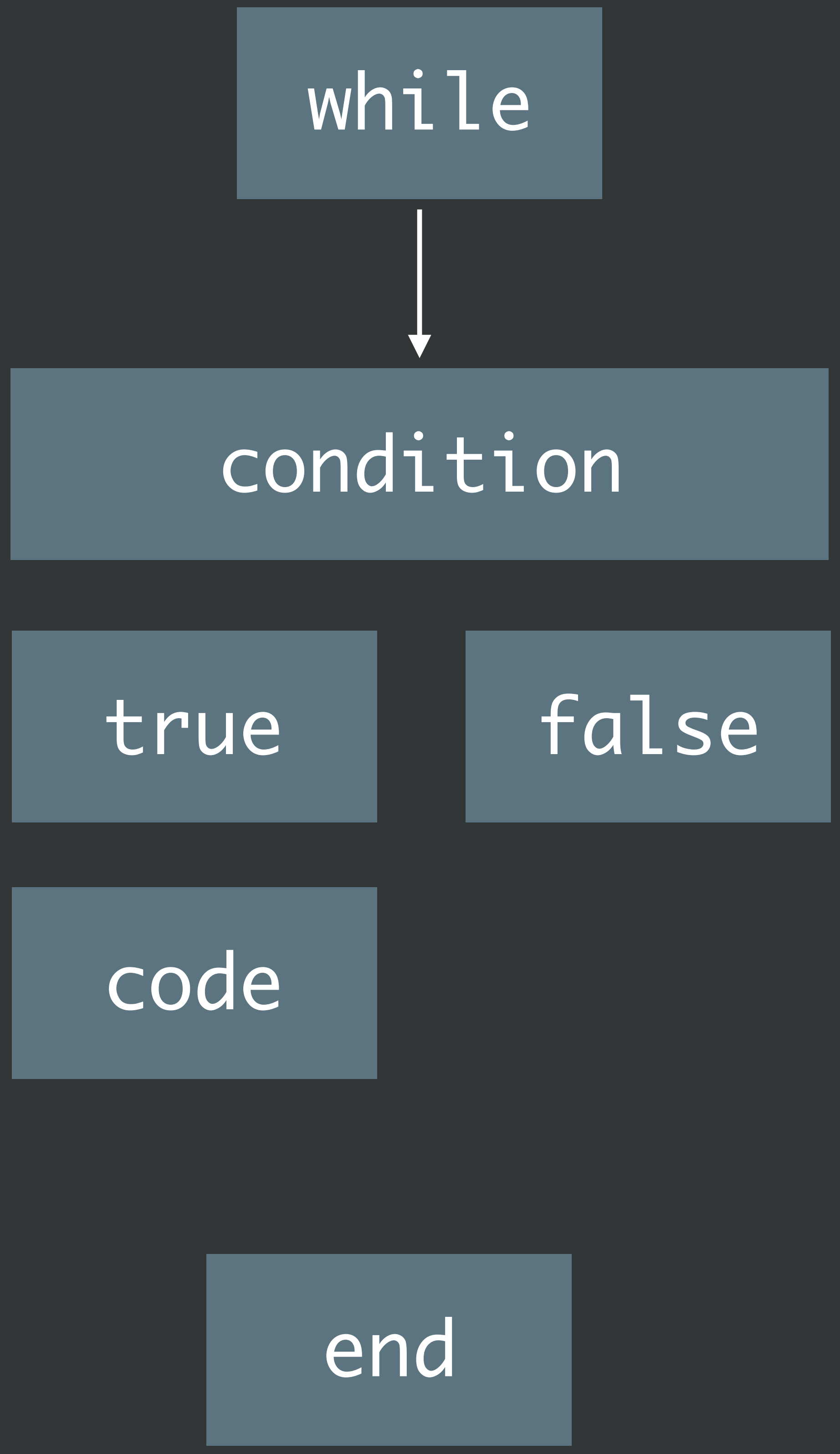
if neither condition is true, our program will evaluate ‘else’ statement, or continue to run if there’s no else statement provided.

ĆWICZENIE: Napisz funkcję, która wyprintuje “Bu!” tylko i wyłącznie jeśli zdefiniowany wcześniej x jest typu float i jego wartość jest większa od 10, a jeśli spełnia jeden z tych warunków wyprintuje “SO CLOSE!”

Napisz funkcję, która wyprintuje “Bu!” tylko i wyłącznie jeśli zdefiniowany wcześniej x jest typu float i jego wartość jest większa od 10, a jeśli spełnia jeden z tych warunków wyprintuje “SO CLOSE!”

```
In: x = 35.0
In: if type(x) == float and x > 10:
      print(‘Bu!’)
    elif type(x) == float or x > 10:
      print(‘SO CLOSE!’)
```

while loops



while

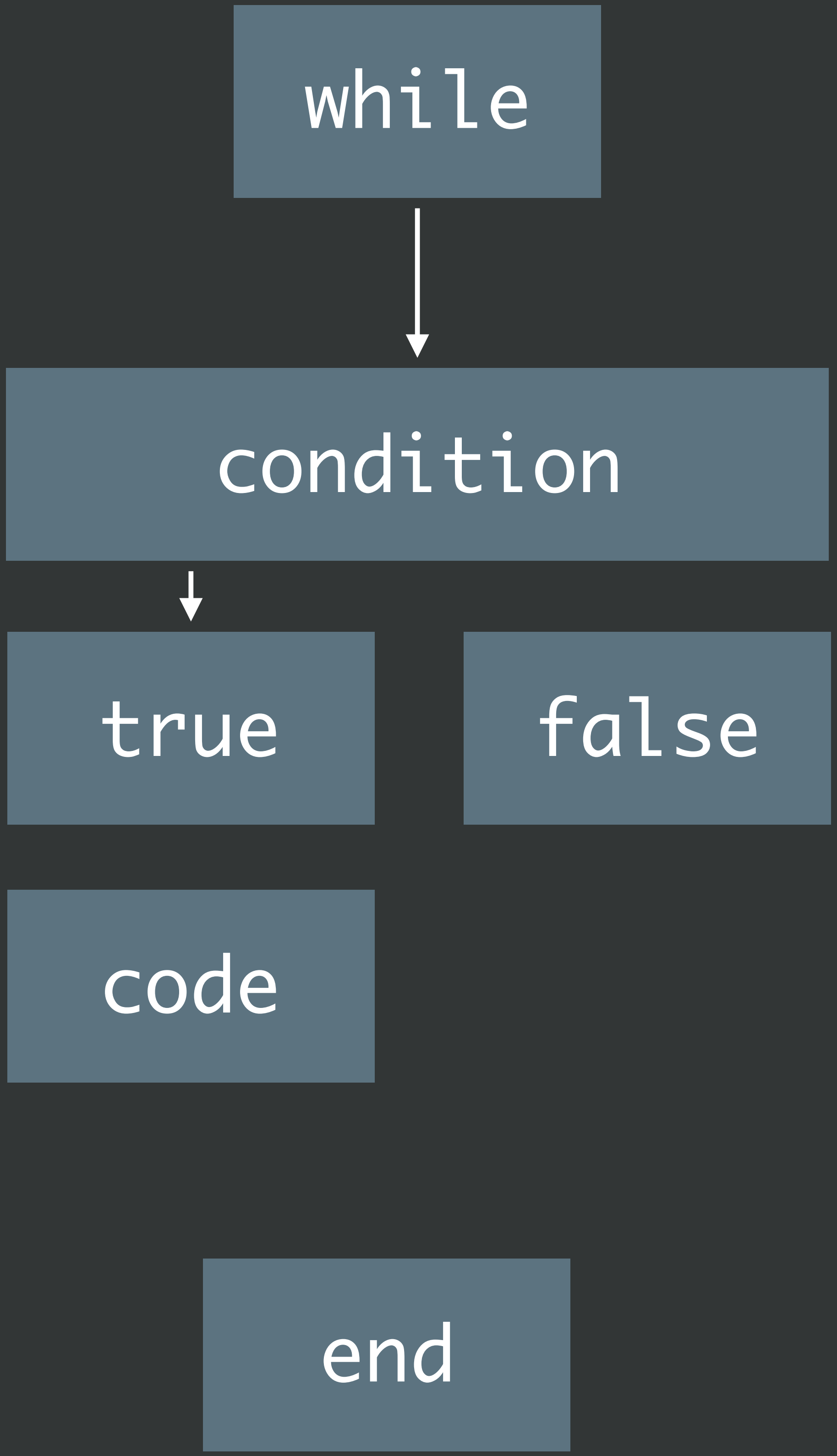
condition

true

false

code

end



while

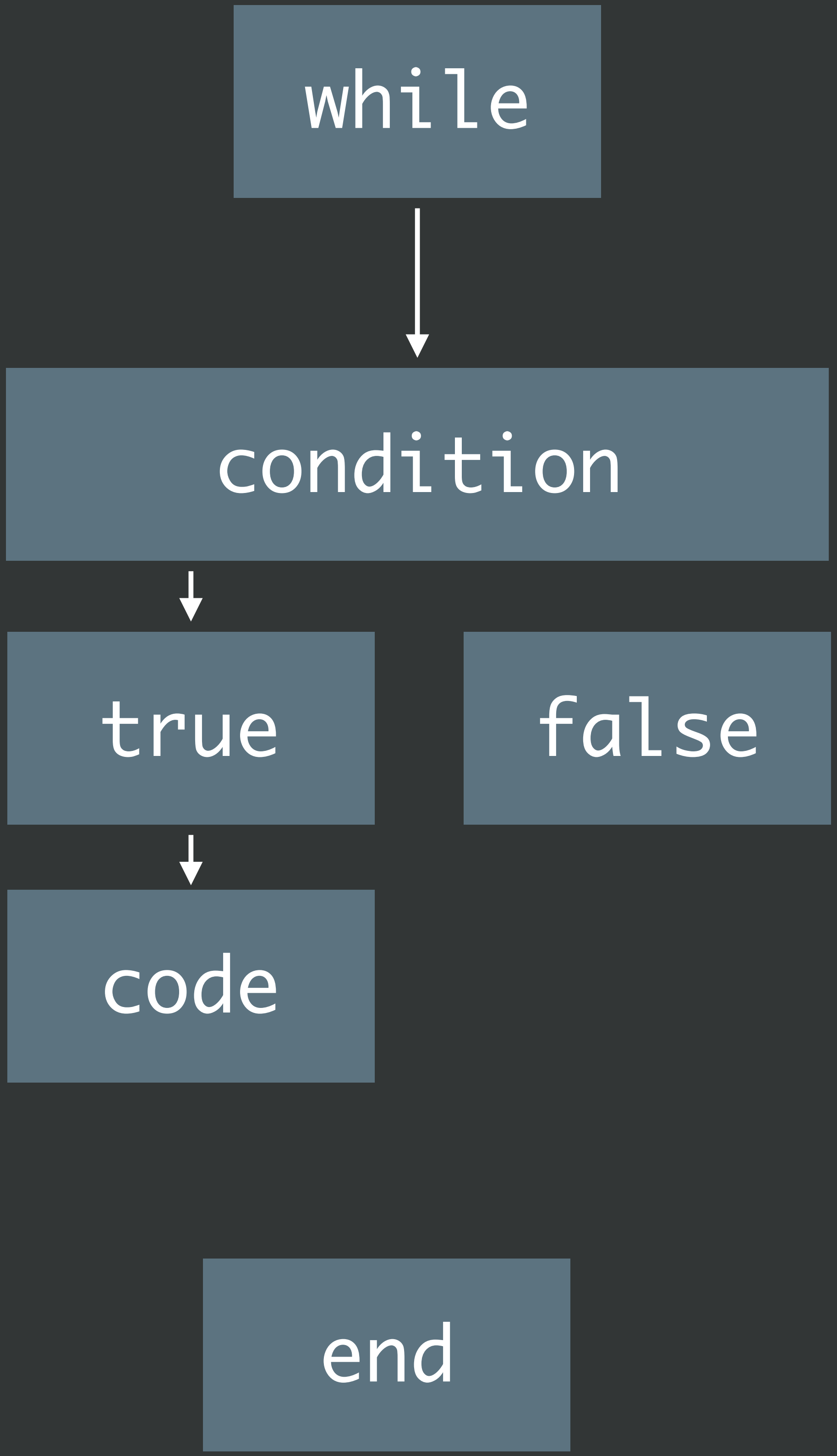
condition

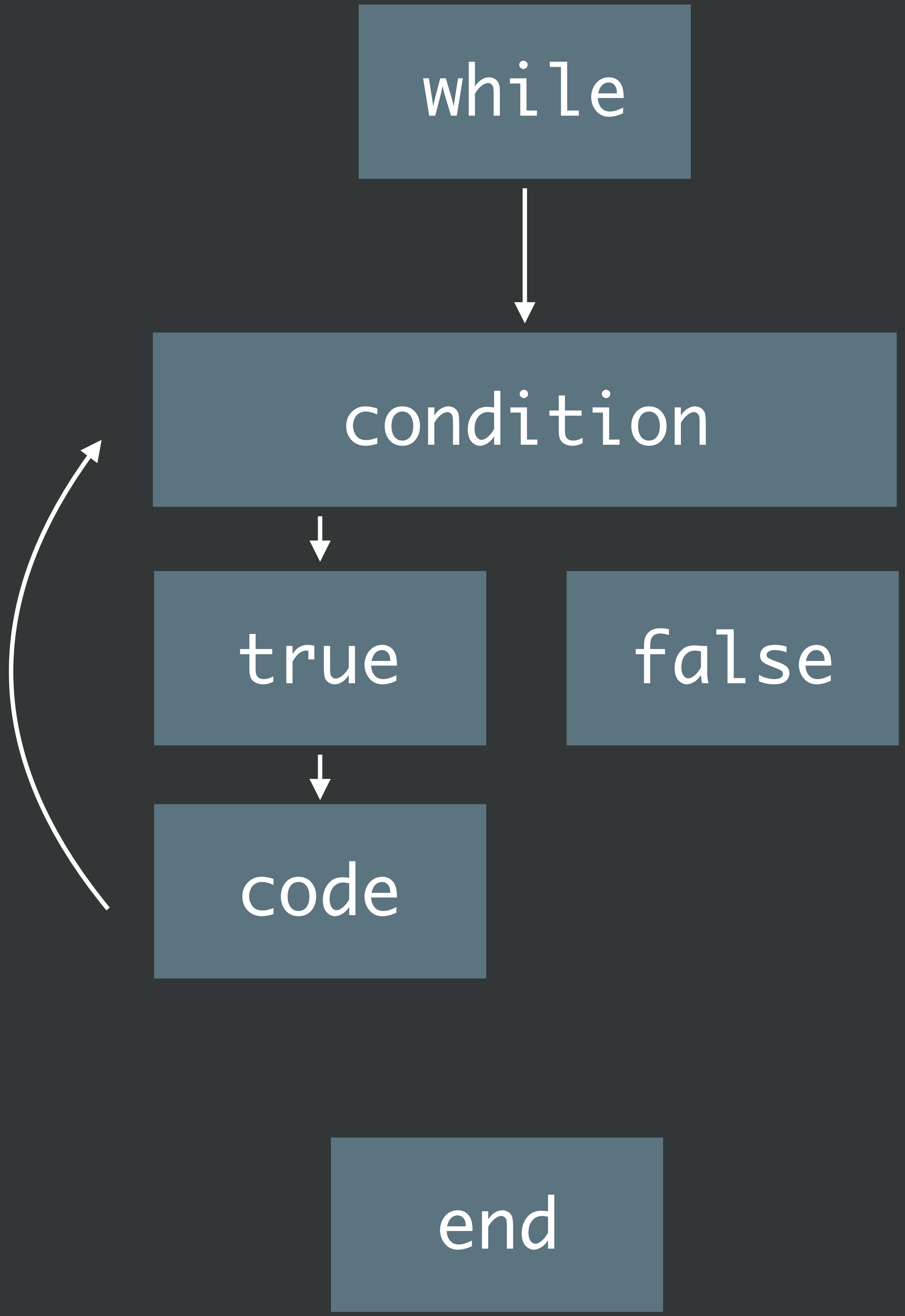
true

false

code

end


while
condition
true
false
code
end



while

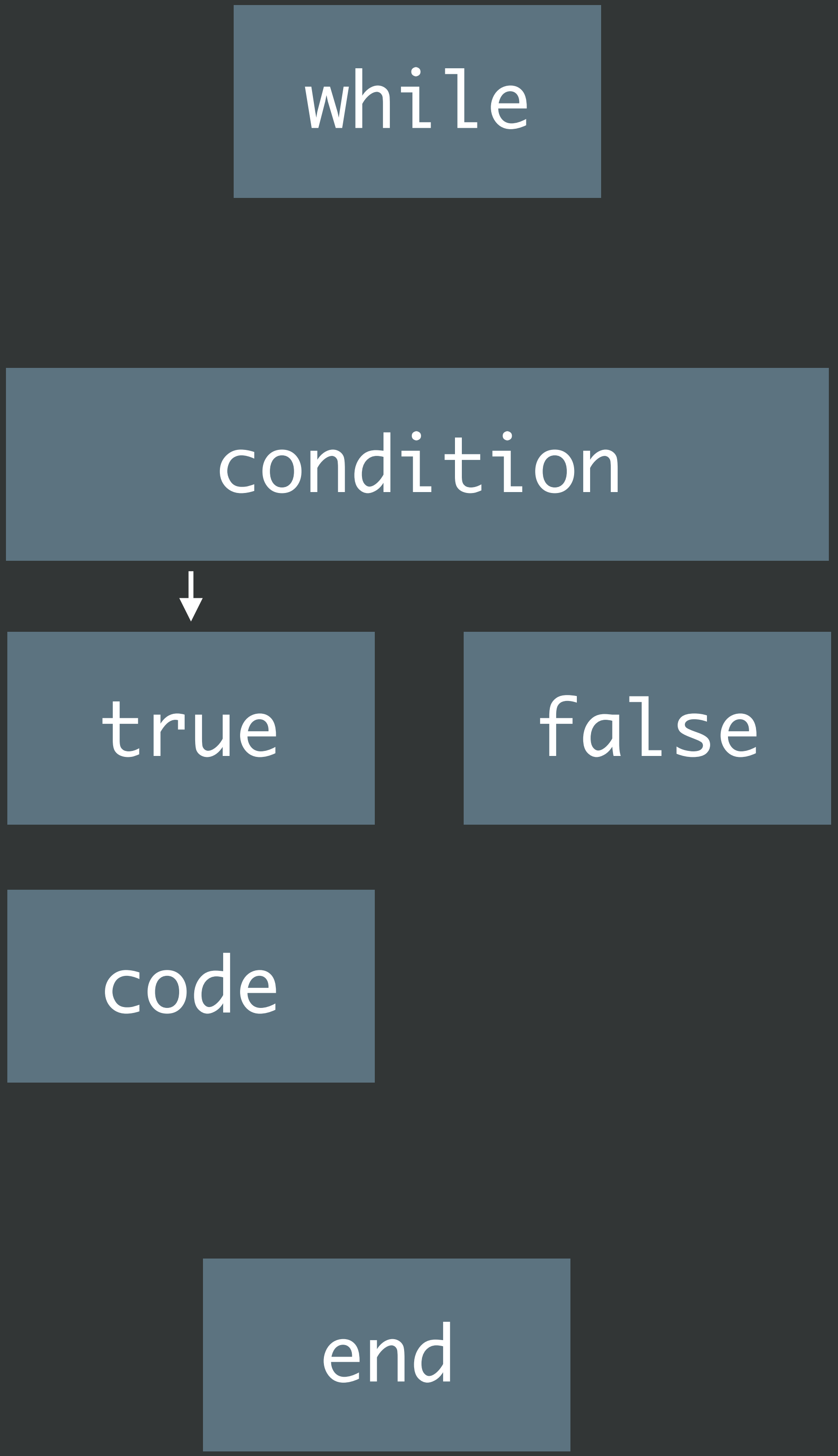
condition

true

false

code

end



while

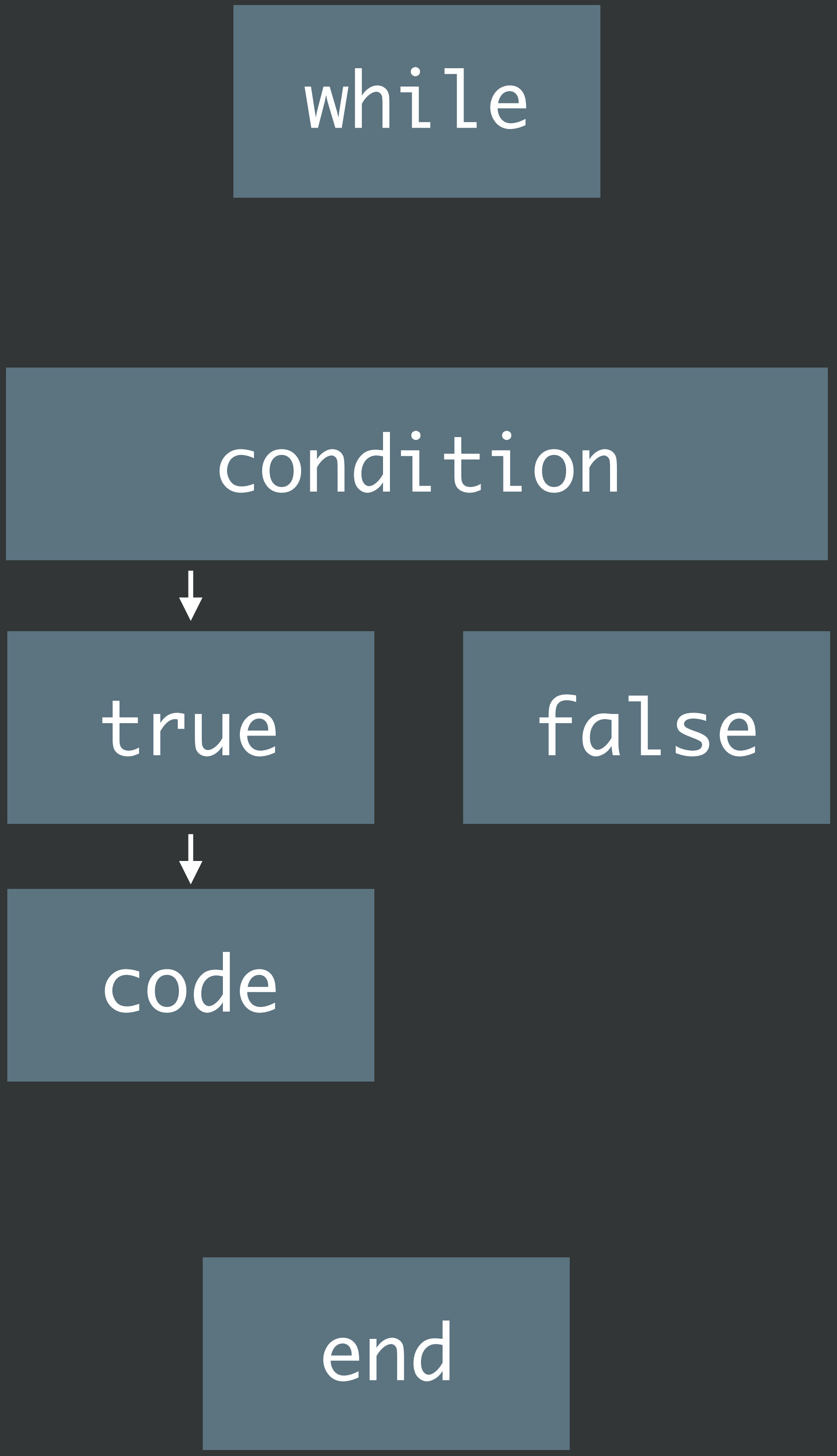
condition

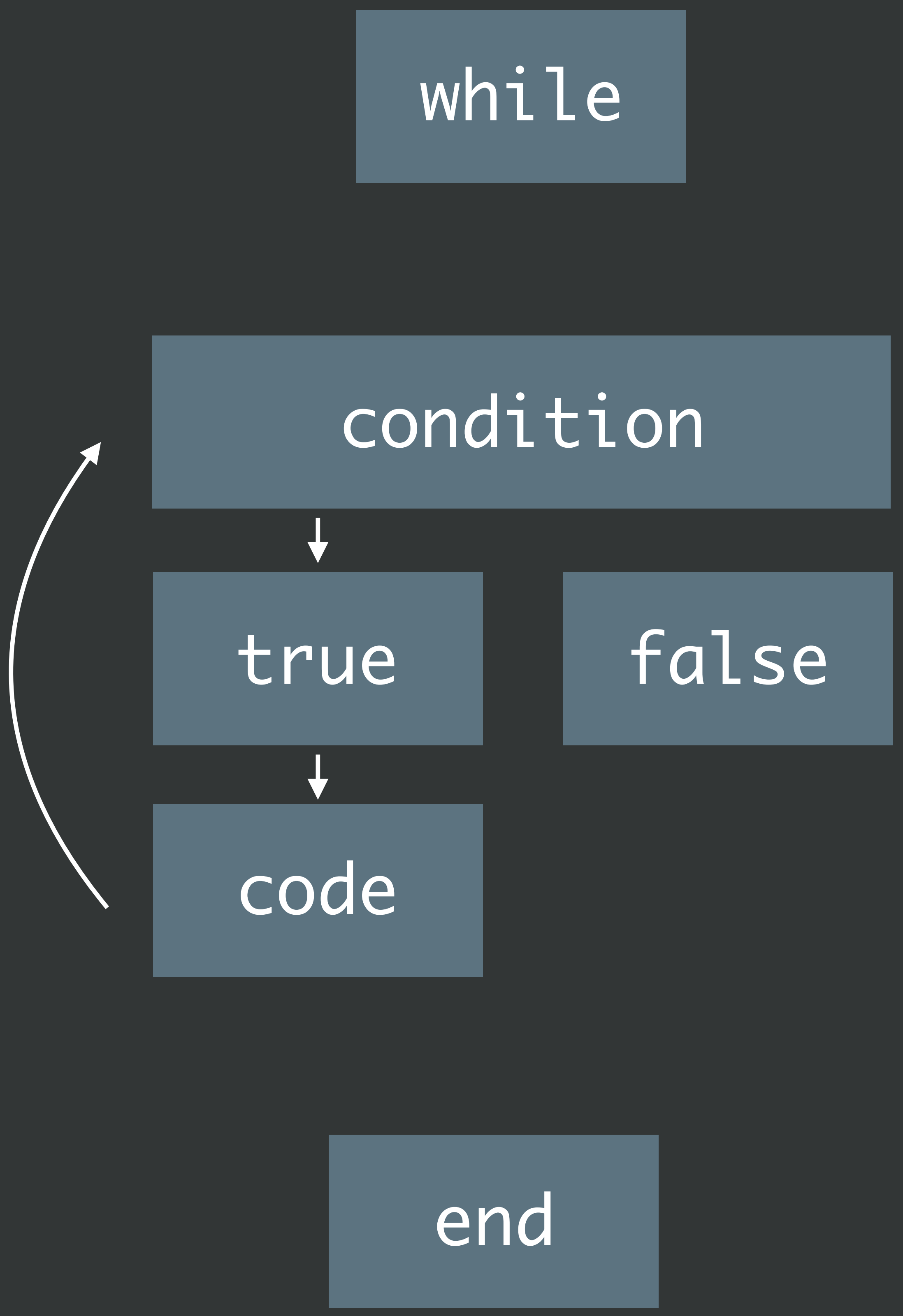
true

false

code

end


while
condition
true
false
code
end



while

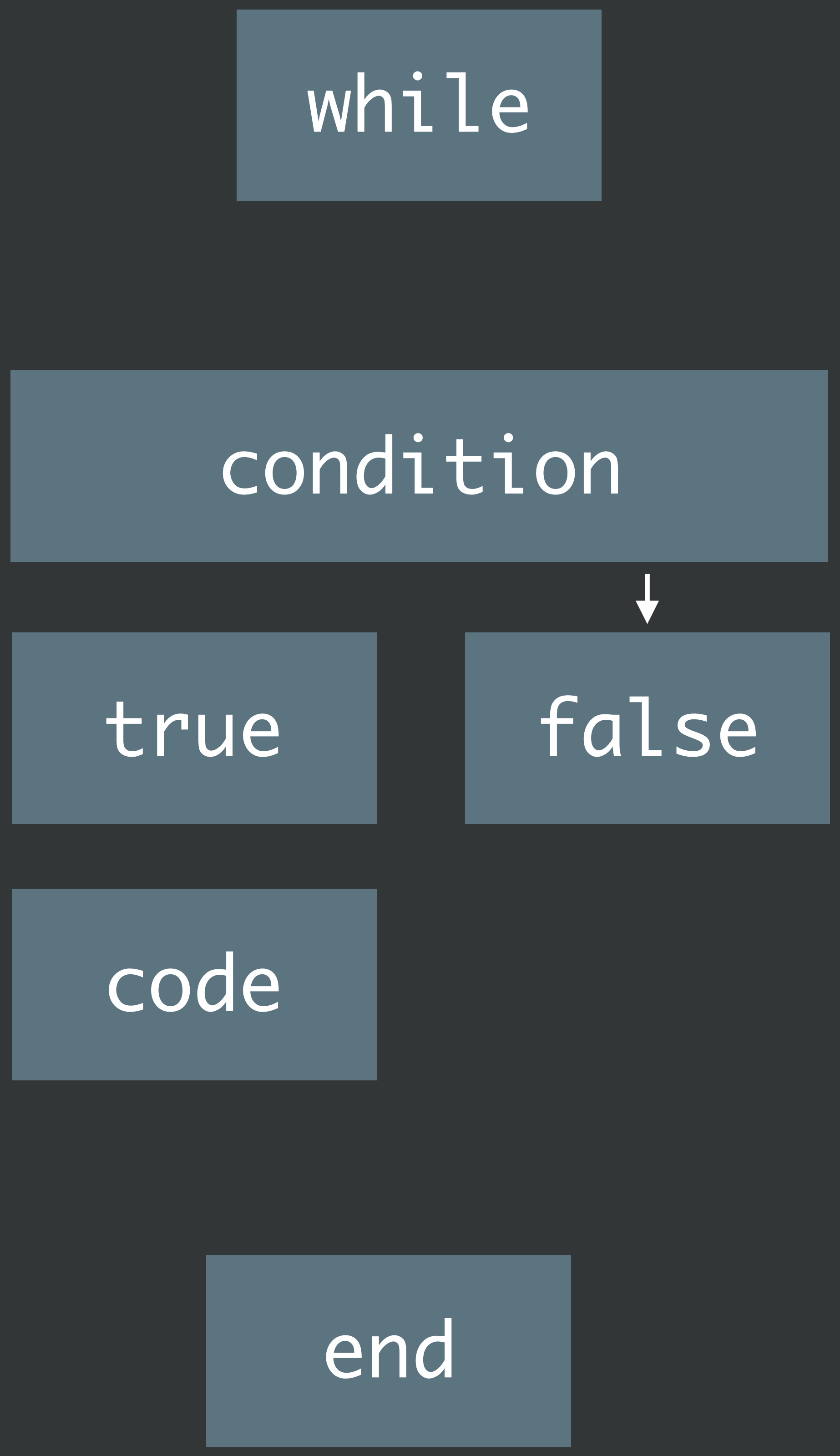
condition

true

false

code

end



while

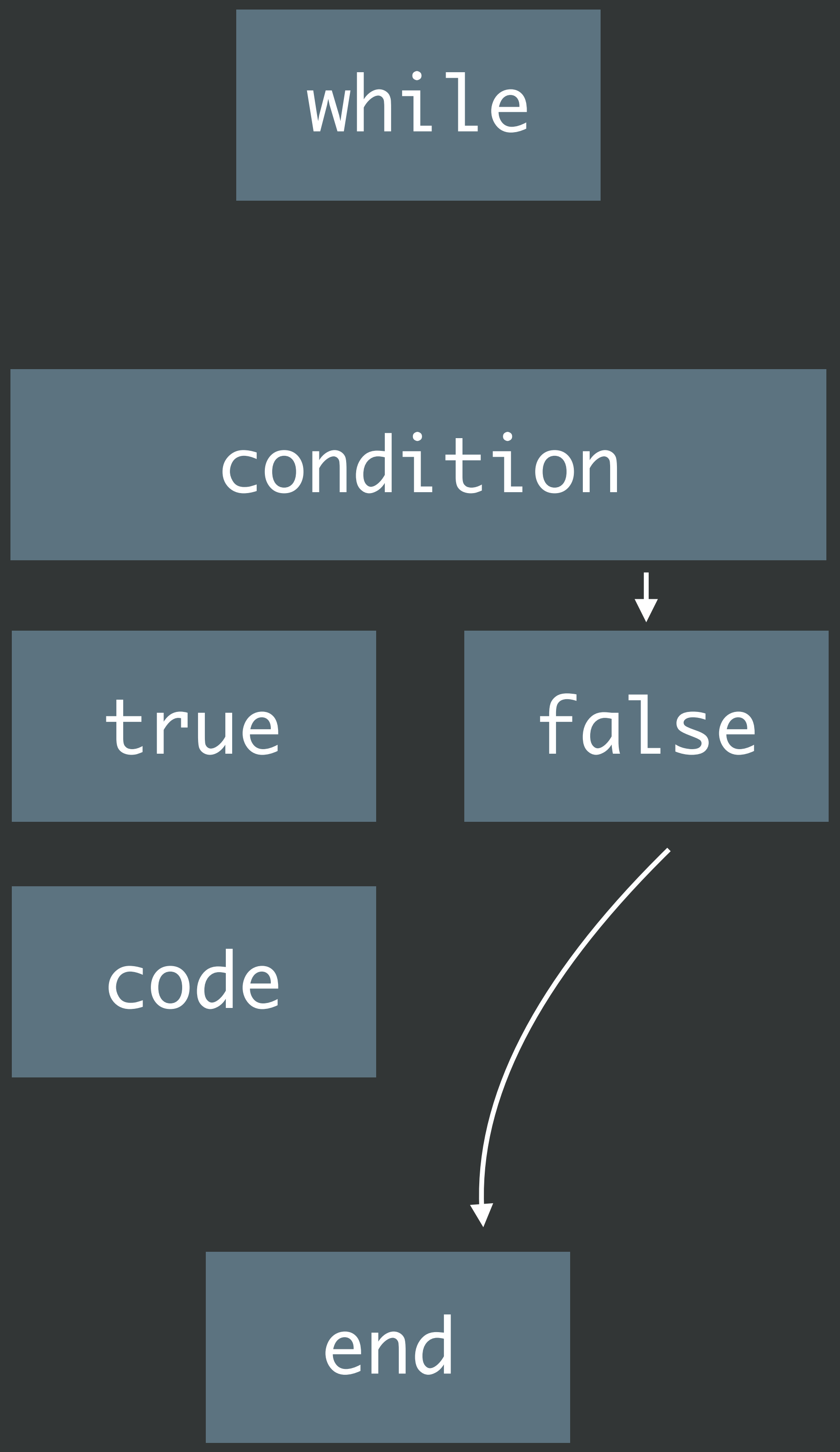
condition

true

false

code

end



while

condition

true

false

code

end

```
In: x = 0
In: while x < 5:
     print(‘Current x is: ‘ + str(x))
     x += 1
```

```
In: x = 0
In: while x < 5:
     print(‘Current x is: ‘ + str(x))
     x += 1

Out: ‘Current x is 0’
Out: ‘Current x is 1’
Out: ‘Current x is 2’
Out: ‘Current x is 3’
Out: ‘Current x is 4’
```

```
In: x = 0
In: while x < 5:
     print(‘Current x is: ‘ + str(x))
     x += 1
In: print(‘out of loop! x is ‘ + str(x))
```

```
In: x = 0
In: while x < 5:
     print(‘Current x is: ‘ + str(x))
     x += 1
In: print(‘out of loop! x is ‘ + str(x))

Out: ‘Current x is 0’
Out: ‘Current x is 1’
Out: ‘Current x is 2’
Out: ‘Current x is 3’
Out: ‘Current x is 4’
Out: ‘out of loop! x is 5’
```

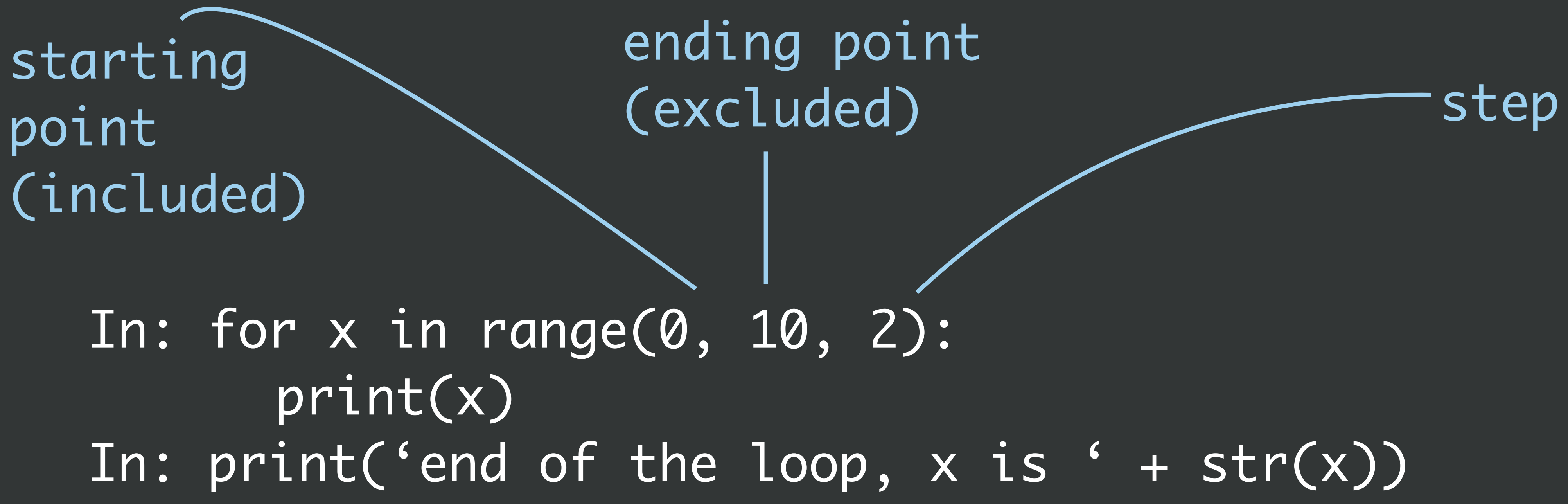
for loops

```
In: for x in range(5):
      print(x)
In: print(‘end of the loop, x is ‘ + str(x))
```

```
In: for x in range(5):
      print(x)
In: print(‘end of the loop, x is ‘ + str(x))

Out: 1
Out: 2
Out: 3
Out: 4
Out: ‘end of the loop, x is 4’
```

```
In: for x in range(0, 10, 2):
      print(x)
In: print(‘end of the loop, x is ‘ + str(x))
```



starting point (included)

ending point (excluded)

step

```
In: for x in range(0, 10, 2):
       print(x)
In: print(‘end of the loop, x is ‘ + str(x))
```

```
In: for x in range(0, 10, 2):
        print(x)
In: print(‘end of the loop, x is ‘ + str(x))

Out: 0
Out: 2
Out: 4
Out: 6
Out: 8
Out: ‘end of the loop, x is 8’
```

```
In: for letter in ‘hello’:
      print(letter)
```

```
In: for letter in ‘hello’:
        print(letter)


Out: h
Out: e
Out: l
Out: l
Out: o
```

breaking from the loop

```
In: for letter in ‘cigarette’:
      if letter == ‘e’:
          break
      print(letter)

In: print(‘the last letter was ‘ + letter)
```

```
In: for letter in ‘cigarette’:
         if letter == ‘e’:
               break
         print(letter)

In: print(‘the last letter was ‘ + letter)

Out: c
Out: i
Out: g
Out: a
Out: r
Out: e
Out: the last letter was e
```

wiedzmy

```
In: x = 1
In: while x < 10:
     if x % 4 == 0:
         break
     print(x)
     x += 1

In: print(‘the last letter was ‘ + str(x))
```

```
In: x = 1
In: while x < 10:
     if x % 4 == 0:
         break
     print(x)
     x += 1

In: print(‘the last x was ‘ + str(x))

Out: 1
Out: 2
Out: 3
Out: 4
Out: the last x was 4
```

exercises

1. Napisz program, który zwraca ciąg Fibonacciego od 1 do granicy wyznaczonej przez dowolną liczbę x.
Np. dla x = 8, program powinien wyprintować 1, 1, 2, 3, 5, 8
(wzór na n-ty wyraz ciągu: $a_n = a_{n-1} + a_{n-2}$)

2. Napisz program, który skończony ciąg geometryczny dla następujących parametrów: a - wyraz początkowy; q - iloraz ciągu; g - granica ciągu

3. Napisz program, który dla określonych parametrów ciągu geometrycznego a - wyraz początkowy, q - iloraz ciągu, n - poszukiwany wyraz, zwróci n-ty wyraz ciągu