Language Basics

Check all syntactically correct statements.

1.) Which input statements are correct?

- ☐ `a = raw_input()`
- ☐ `a = raw_input(“enter a number”)`
- ☐ `a = raw_input(enter your name)`

2.) Which print statements are correct?

- ☐ print “9” + “9”
- ☐ print int(“nine”)
- ☐ print str(9) + “nine”
- ☐ print 9 + 9
- ☐ nine = 9
  print nine

3.) Which are correct arithmetical operations?

- ☐ `a = 1 * 2`
- ☐ `2 = 1 + 1`
- ☐ `5 + 6 = y`
- ☐ `seven = 3 * 4`

4.) Which are correct variable names?

- ☐ `result`
- ☐ `my.result`
- ☐ `print`
- ☐ `result77`

5.) Which are correct type conversions?

- ☐ `int(7.0+0.1)`
- ☐ `str(1.2 * 3.4)`
- ☐ `float(“77”+”.0”)`
- ☐ `str( 9 / 0 )`

6.) Which operations result in 8?

- ☐ `65 // 8`
- ☐ `17 % 9`
- ☐ `2 ** 4`
- ☐ `64 ** 0.5`

7.) Which lines are commented?

- ☐ `“””This is a comment”””`
- ☐ `# This is a comment`
- ☐ `// this is a comment`
- ☐ `'''This is a comment'''`

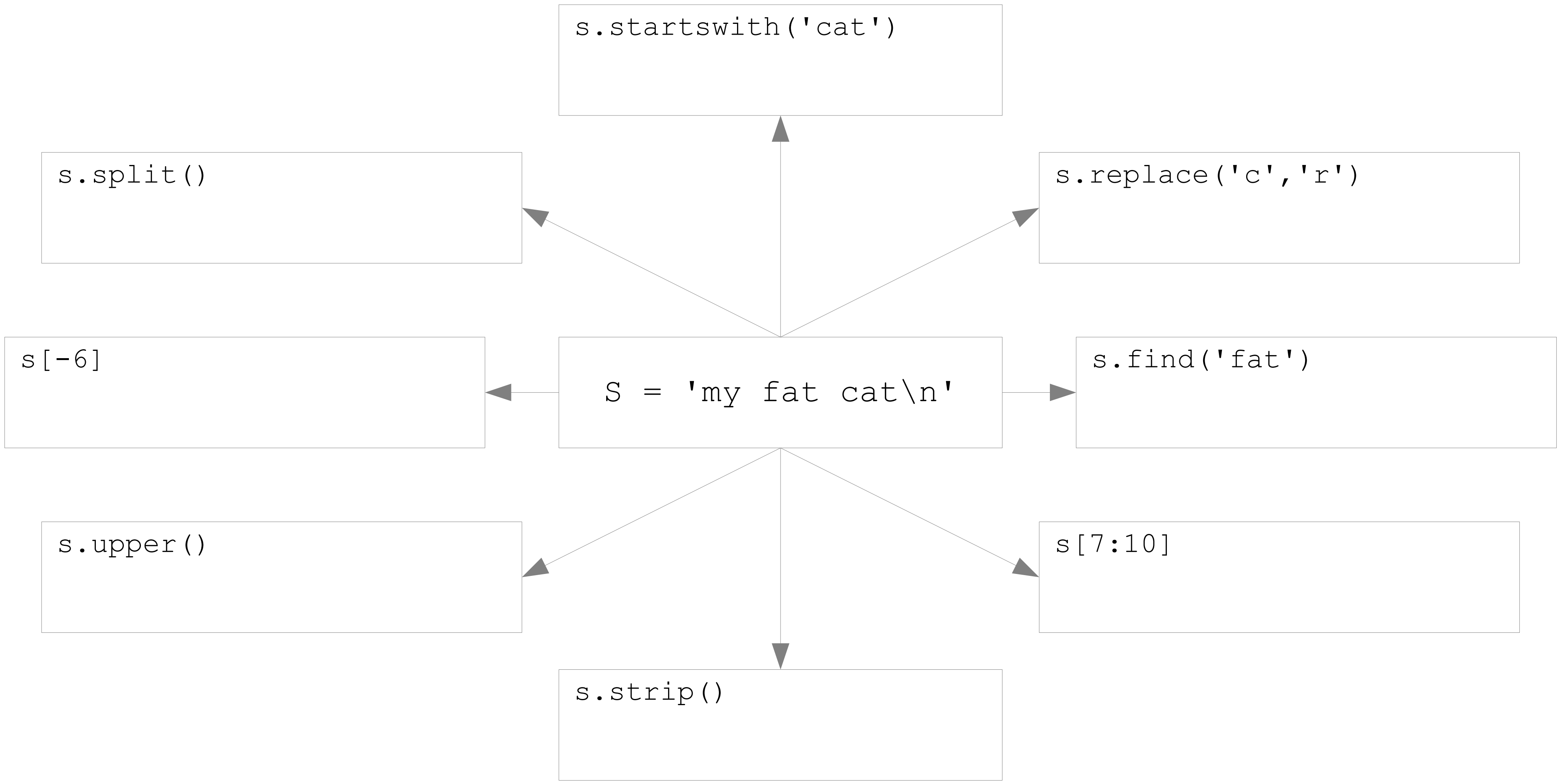
Data Types

Find the matching pairs of expressions and values.

`1023`	`boolean`
`None`	`int`
`[2, 4, 8, 16]`	`tuple`
`True`	`list`
`17.54`	`str`
`('Roger', 1952)`	`dict`
`“my fat cat”`	`NoneType`
`{'name':'Roger', 'birth':1952}`	`float`

Manipulating Strings

Write the result of each operation into the fields.



```
S = 'my fat cat\n'
```

```
s.startswith('cat')
s.split()
s.replace('c','r')
s[-6]
s.find('fat')
s.upper()
s[7:10]
s.strip()
```

K. Rother, A. Philips
& K. Milanowska

String formatting

Find the matching pairs of expressions and values.

expressions	values
`"R.%s"%("GB")`	` 3.  RGB`
`"%2i. %4s"%(3,"RGB")`	` 3.14`
`"%5.2f"%3.1415`	`R.GB`
`'%s\t%s\tRG'%(B,'\t')`	`RG   7B`
`"%5i"%(3.1415)`	`    3`
`"RG%4iB"%(7)`	`B              RG`

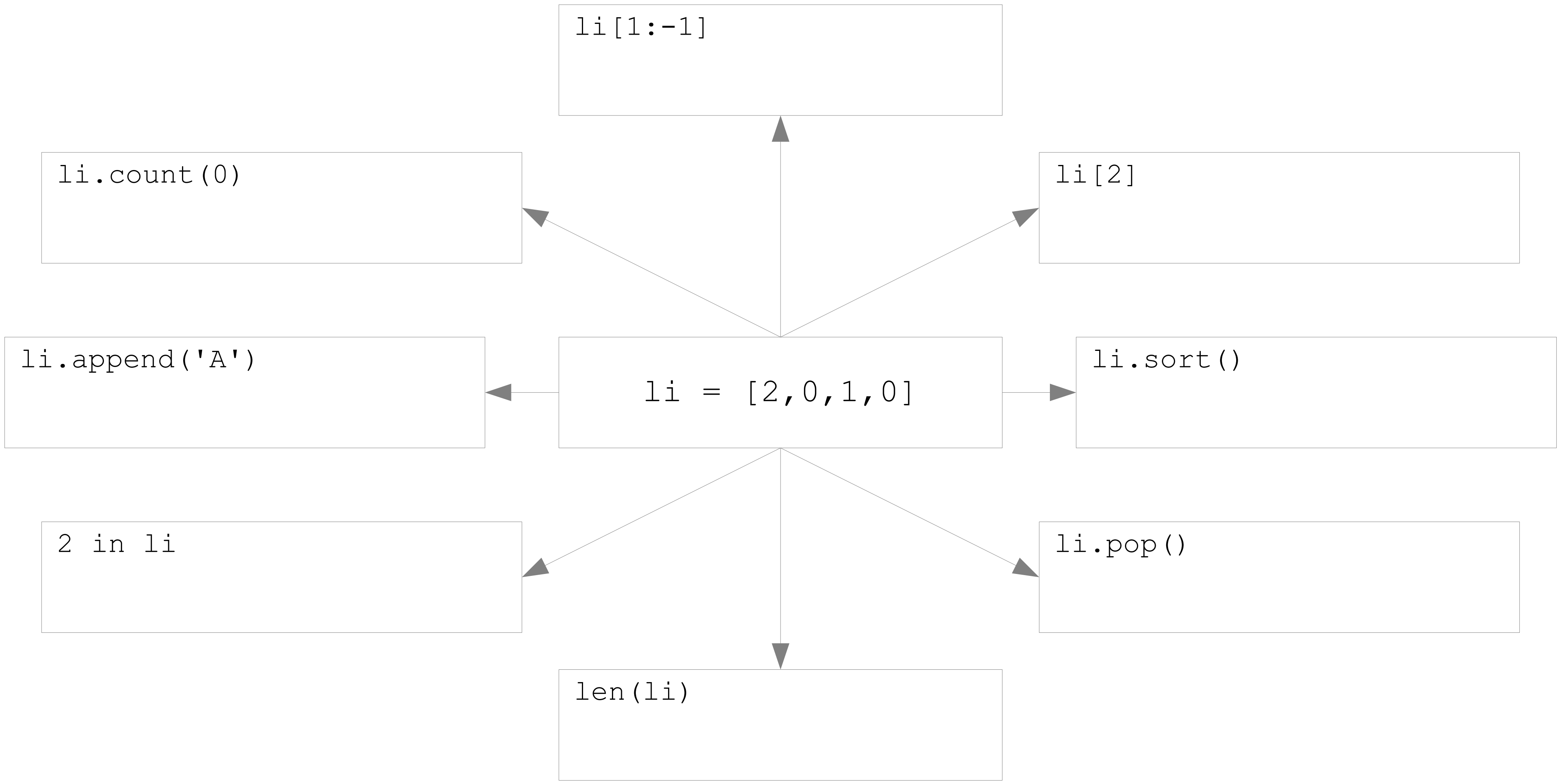
Functions and Modules

Find the matching pairs.

An optional parameter	Tells about properties of anything in Python.
An import statement	May return several values as a tuple.
A package	Carries the risk of an endless loop.
The dir() function	Is a directory with Python modules.
Every package	Is best written on top of a file.
A function that calls itself	Must contain a file __init__.py
A function in Python	Must be written after obligatory parameters.
A function	May modify its own parameters.

Manipulating Lists

Write the result of each operation into the fields.



```
li = [2,0,1,0]
```

```
li[1:-1]
```

```
li.count(0)
```

```
li[2]
```

```
li.append('A')
```

```
li.sort()
```

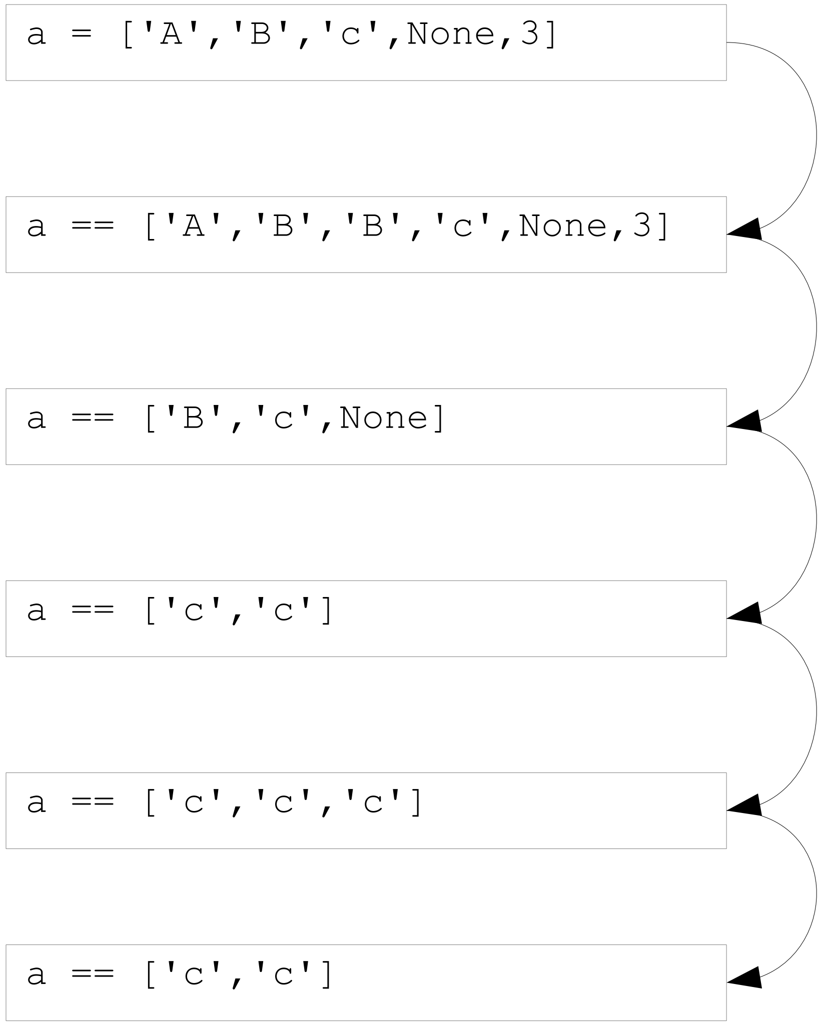
```
2 in li
```

```
li.pop()
```

```
len(li)
```

Functions operating on lists (1)

Write the correct operations to the arrows.

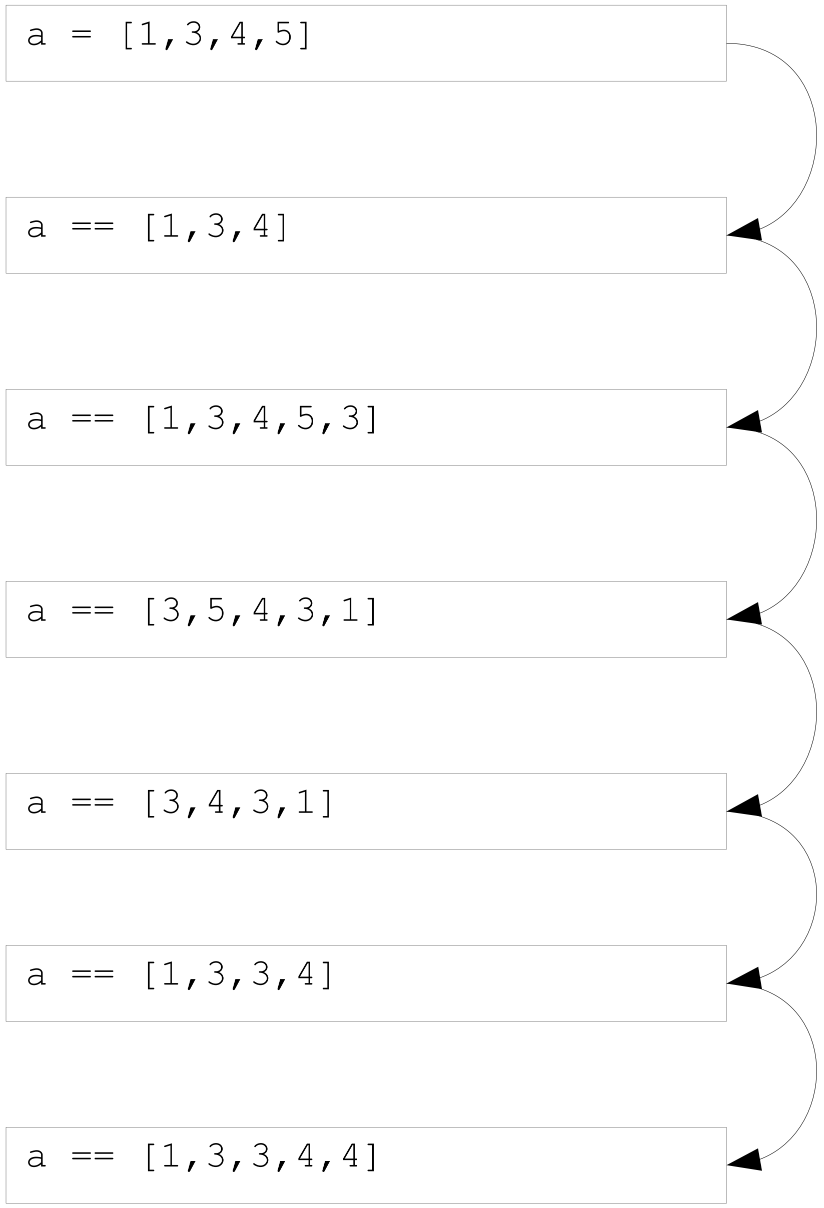


```
a = ['A','B','c',None,3]
a == ['A','B','B','c',None,3]
a == ['B','c',None]
a == ['c','c']
a == ['c','c','c']
a == ['c','c']
```

1	`a = a[2:5]`
2	`a = [a[-2]] + [a[1]]`
3	`a = a[:2]`
4	`a = [a[-1]]*3`
5	`a = a[:2] + a[1:]`

Functions operating on lists (2)

Write the correct operations to the arrows.



```
a = [1,3,4,5]
a == [1,3,4]
a == [1,3,4,5,3]
a == [3,5,4,3,1]
a == [3,4,3,1]
a == [1,3,3,4]
a == [1,3,3,4,4]
```

1	`a.reverse()`
2	`a.sort()`
3	`a.pop()`
4	`a.append(4)`
5	`a = a + [5,3]`
6	`a.remove(5)`

K. Rother, A. Philips
& K. Milanowska

Functions operating on lists (3)

Write the result of each operation into the fields.

```
a = ['a','b','c']
b = [10,12,14,16]
```

```
sum(b)
```

```
sum(range(5))
```

```
range(3)
```

```
range(5,8)
```

```
range(6,0,-1)
```

```
for i,j in enumerate(a):
    print i,j
```

```
zip(a,b)
```

Manipulating Lists

Check the correct answer.

1.) What does the list *b* contain?

```
a = [8,7,6,5,4]
b = a[2:4]
```

- ☐ `[7,6,5]`
- ☐ `[7,6]`
- ☐ `[6,5]`
- ☐ `[6,5,4]`

2.) Which of the following code pieces results in

```
a == [2,4,6]
```

- ☐ `a = [1,2,3] * 2`
- ☐ `a = [int(s) for s in “246”]`
- ☐ `a = [x*2 for x in range(3)]`
- ☐ `a = [2**1]+[2**2]+[2**3]`

Working with Tuples

Check all correct answers.

1.) Which are correct tuples?

- ☐ `( 1, 2, 3)`
- ☐ `(“Jack” “Knife”)`
- ☐ `('blue', [0,0,255])`
- ☐ `[ 1, “word” ]`

2.) What are tuples good for?

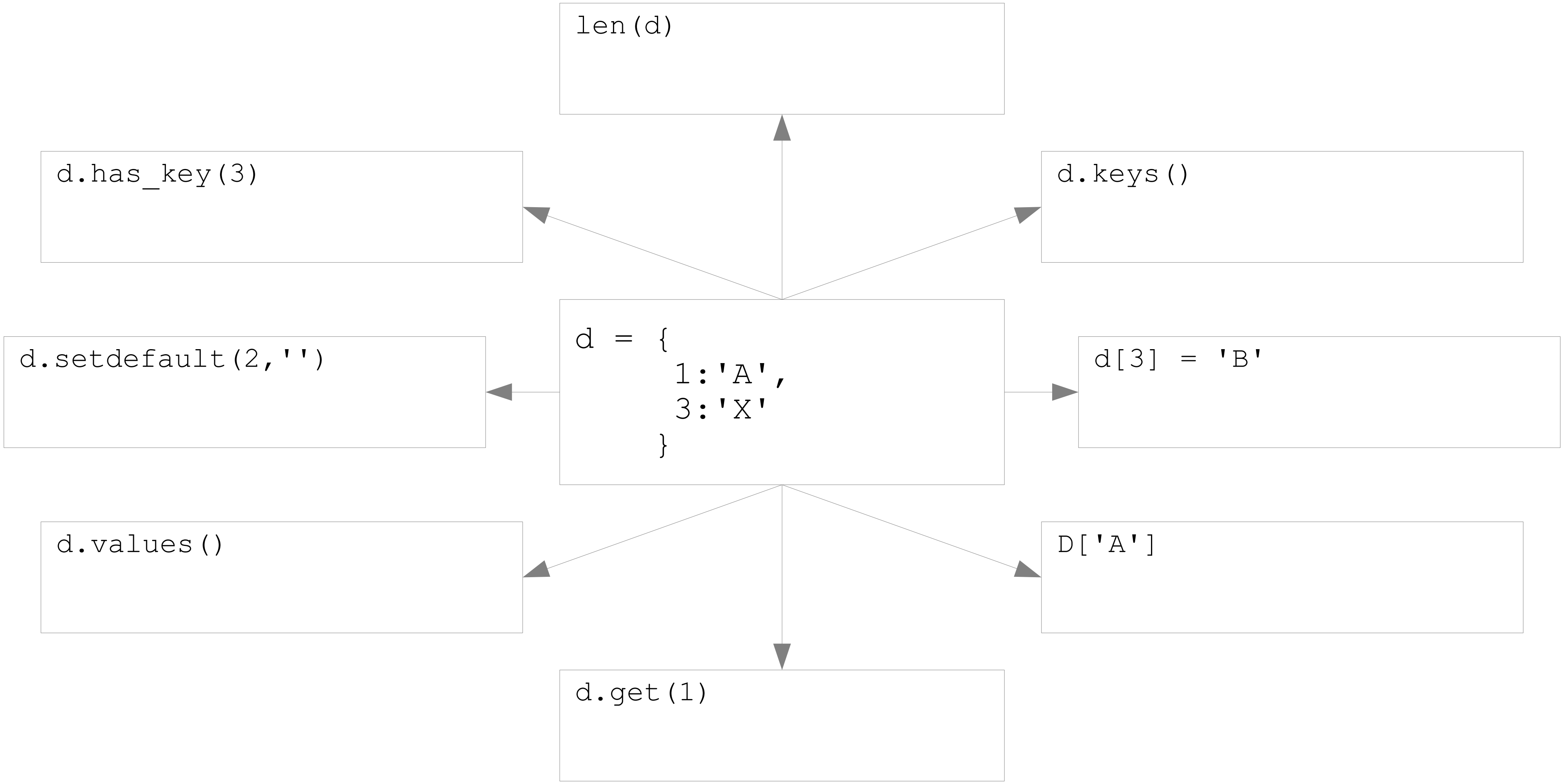
- ☐ `Grouping data.`
- ☐ `Managing values that change.`
- ☐ `Running a for loop over them.`
- ☐ `Sorting.`

3.) On what data types does the len() function work on?

- ☐ `lists`
- ☐ `dictionaries.`
- ☐ `strings.`
- ☐ `tuples.`

Manipulating Dictionaries

Write the result of each operation into the fields.



```
d = {
     1:'A',
     3:'X'
    }
```

```
len(d)
```

```
d.has_key(3)
```

```
d.keys()
```

```
d.setdefault(2,'')
```

```
d[3] = 'B'
```

```
d.values()
```

```
D['A']
```

```
d.get(1)
```

Manipulating Dictionaries

Check the correct answer.

1.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
print d['A']
```

- ☐ `False`
- ☐ `“B”`
- ☐ `True`
- ☐ `1`

2.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
print d.has_key('B')
```

- ☐ `1`
- ☐ `True`
- ☐ `'B'`
- ☐ `False`

3.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
print d.values()
```

- ☐ `True`
- ☐ `['A',1,True]`
- ☐ `3`
- ☐ `[1,'B','A']`

4.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
print d.keys()
```

- ☐ `[1,'B','A']`
- ☐ `['A','B',1]`
- ☐ `[1,'A','B']`
- ☐ The order may vary.

5.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
print d['C']
```

- ☐ `None`
- ☐ `'C'`
- ☐ `an Error`
- ☐ False

6.) What do these commands produce?

```
d = {1:'A', 'B':1, 'A':True}
d.setdefault('C',3)
print d['C']
```

- ☐ `3`
- ☐ `“C”`
- ☐ `None`
- ☐ `an Error`

Loops and conditional statements

Check the correct statements.

1.) Which of these while commands are correct?

☐ `while a = 1:`

☐ `while a+7:`

☐ `while len(c)>10:`

☐ `while a and (b-2 == c):`

☐ `while b==1`

☐ `while s.find('c')>=0:`

2.) Which of these statements are correct?

☐ 'while' is also called a conditional loop.

☐ The expression after while may contain function calls.

☐ It is possible to write endless while loops.

☐ The colon after while may be omitted.

☐ The code block after while is executed at least once.

3.) What are possible structures of a conditional statement?

☐ if <expression> .. else

☐ if <expression> .. else if <expression>

☐ if <expression> .. elif <expression> .. else

☐ if <expression> .. elif <expression> ..else <expression>

☐ If <expression>.. else <expression> .. efli

4.) Which of these for commands are correct?

☐ `for char in “ABCD”:`

☐ `for i in range(10):`

☐ `for num in (4,6,8):`

☐ `for k in 3+7:`

☐ `for (i=0; i<10; i+=1):`

☐ `for var in seq:`

5.) Which of these if statements are syntactically correct?

☐ `if a and b:`

☐ `if len(s) == 23:`

☐ `if a but not b<3:`

☐ `if a ** 2 >= 49:`

☐ `if a != 3:`

☐ `if (a and b) or (c and d):`

K. Rother, A. Philips
& K. Milanowska

Modules and Packages

Check the correct answer(s).

1.) Which of these import statements are correct?

- [ ] `import re`
- [ ] `import re.sub`
- [ ] `from re import sub`
- [ ] `from re.sub import *`
- [ ] `from .re import *`
- [ ] `from re import *`

2.) Where does Python look for modules to import

- [ ] In the sys.path variable.
- [ ] In the current working directory.
- [ ] In the directory where the current module is.
- [ ] In the directory where Python was started.
- [ ] In the site-packages folder
- [ ] In directories in the PYTHONPATH variable
- [ ] In the root directory.

3.) Which statements about packages are true?

- [ ] A package is a directory with modules.
- [ ] A package may contain zero modules.
- [ ] Packages in site-packages/ are imported automatically.
- [ ] A package must contain a __init__.py file.
- [ ] A package may contain no code.
- [ ] Packages are useless in small programs.

4.) Which packages are installed by default?

- [ ] `os` – manipulating files and directories.
- [ ] `psyco` – makes Python faster
- [ ] `time` – accessing date and time.
- [ ] `csv` – reads and writes tables.
- [ ] `numpy` – number crunching.
- [ ] `pdb` – Python debugging.

K. Rother, A. Philips
& K. Milanowska

While loops

Match the expressions for the while loops run the designated number of times

```
a = 5
while [  ] :
     a = a -1
```

5x

```
a = 2
while [  ] :
     a += 4
```

5x

```
a = 2
while [  ] :
     a = -a * 2
```

2x

```
a = 7
while [  ] :
     a -= 2
```

4x

`a != 0`

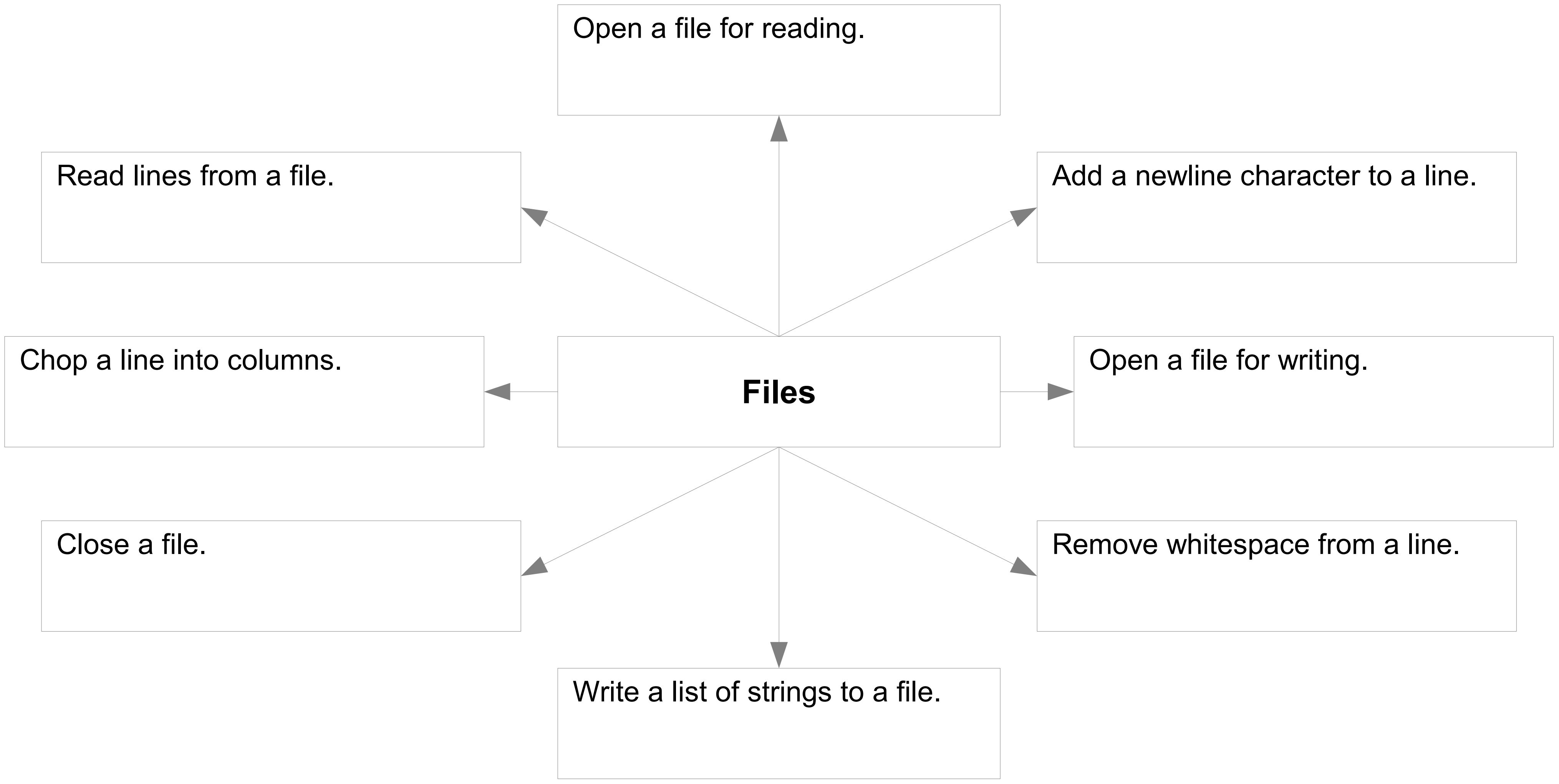
`a >= 0`

`a < 19`

`abs(a) < 7`

Reading and Writing Files

Write Python commands into the fields.


Open a file for reading.
Read lines from a file.
Add a newline character to a line.
Chop a line into columns.
Files
Open a file for writing.
Close a file.
Remove whitespace from a line.
Write a list of strings to a file.

Error Handling

Check all correct answers.

1.) Which commands result in an Exception?

- ☐ `f = open(“:::”)`
- ☐ `char = “abc”[7]`
- ☐ `a = (5+9) / (6-(2*3))`
- ☐ `num = [1,2,3][0]`
- ☐ `l = range(10)[:20]`
- ☐ `num = {1:'a'}['a']`

2.) Which are common types of Exceptions?

- ☐ `ZeroDivisionError`
- ☐ `IOError`
- ☐ `ValueError`
- ☐ `NullPointerException`
- ☐ `KeyError`
- ☐ `InfiniteLoopError`

3.) Which commands for managing Exceptions exist?

- ☐ `try: … else: ...`
- ☐ `raise ValueError('text')`
- ☐ `try: … except: … error:`
- ☐ `try: … except: ...`

Working with Files

Check all correct answers.

1.) Which are correct commands working with files?

- ☐ `for line in open(filename):`
- ☐ `f = open(filename,'w')`
- ☐ `open(filename).writelines(out)`
- ☐ `f.close()`

2.) Which statements about the csv module are correct?

- ☐ `It can save tables of strings and numbers.`
- ☐ `csv reads tables of strings.`
- ☐ `csv cannot handle the quote character.`
- ☐ `Files need to have the .csv suffix.`

The math module

Find the matching pairs of functions and values.

functions	values	
`y = math.cos(x)`	`x = 180`	`y == 3.14159..`
`y = math.radians(x)`	`x = 1`	`y == 3.14159..`
`y = math.factorial(x)`	`x = 81`	`y == 9.0`
`y = math.pi * x`	`x = 32`	`y == 5.0`
`y = math.sqrt(x)`	`x = 6.28318..`	`y == 1.0`
`y = math.log(x,2)`	`x = 4`	`y == 24`

The os module

Insert the correct functions into the gaps.

The Python module ☐ is very useful for interactions with the operating system. In fact, it is a combination of two modules: os and ☐. Before any of them can be used, the modules need to be activated by the ☐ statement.
Among the most frequently used operations is the ☐ function, that returns a list of all files in the given directory. If a program already has a filename, but it needs to be checked whether the file really exists, the function ☐ will return **True** or **False**. If a file needs to be deleted, this can be done using ☐.
A very useful feature of the **os.path** module is that it helps operating with directory names. Probably the most frequently used function is ☐, that separates a file name from directory names.
But **os** can do even more: You can use any shell command from a Python program with ☐ - However, this method has disadvantages: it depends on the operating system, and is a potentially insecure.

1 `os.access(fn,os.F_OK)`

2 `os.remove(filename)`

3 `os.path`

4 `os.listdir()`

5 `os.system(command)`

6 `os.path.split(os.getcwd())`

7 `import os`

8 `os`

K. Rother, A. Philips
& K. Milanowska