```
suzan
log
cara
boyana
aglaia
ada
```

Now we will add an `if` statement, with the condition that the variable `name` should start with an "a".

If the condition is met (if it is `True` so to say), then it will continue:

```
In [67]: for name in xpub1:
             if name.startswith("a"):
                 print(name)
```

```
aglaia
ada
```

You can add an `else` statement too, which will catch all the cases that were `False`:

```
In [68…for name in xpub1:
             if name.startswith("a"):
                 print(name)
             else:
                 print("This name does not start with an
       'a':", name)
```

```
This name does not start with an 'a': suzan
This name does not start with an 'a': log
This name does not start with an 'a': cara
This name does not start with an 'a': boyana
aglaia
ada
```

```
In [ ]:
```

Hello Python!

> Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. (https://en.wikipedia.org/wiki/Python_(programming_language))

Where can you work with Python?

- `$ python3`
- script.py + `$ python3 script.py`
- in Jupyter Lab using notebooks

Documentation and resources:

- https://docs.python.org/3/library/index.html
- https://devdocs.io/
- https://pythonfordesigners.com/
- `SHIFT` + `TAB` (to read Python documentation inline)
- `TAB` (to get a list of options inline)

variables

```
In [2]: a = 3
        b = 2
```

```
In [3]: a * b
```

```
Out[3]: 6
```

```
In [4]: b + b
Out[4]: 4
```

You can store the result of an arithmic in another variable, for example:

```
In [5]: c = a + b
In [6]: c
Out[6]: 5
```

print()

You can see the result of your arithmic expression by directly running a cell, or, you can use the built-in print function.

```
In [7]: a + b
Out[7]: 5
In [8]: print(a + b)
5
```

numbers

integers

```
In [9]: type(2)
Out[9]: int
```

```
Out[59]: 'XPUB says Hello!'
In [60]: x.split()
Out[60]: ['XPUB', 'says', 'Hello!']
In [61]: x.startswith("x")
Out[61]: False
In [62]: x.startswith("X")
Out[62]: True
In [63]: x.find("P")
Out[63]: 1
In [64]: x.index("H")
Out[64]: 10
```

if/else statements

Sometimes you want to only do something `if` a condition is met.

For example, we only want to print the names of XPUB1 that start with an "a"...

Let's first make a `for` loop and print all the names:

```
In [65]: for name in xpub1:
             print(name)
```

If we combine that with the f-string we used above:

```
In [5…  print(f"hello { choice(xpub1) }, welcome to my {
        choice(items) }")
```

```
hello boyana, welcome to my library
```

for loop with range()

```
In [5…  for number in range(6):
            print(f"hello { choice(xpub1) }, welcome to
        my { choice(items) }")
```

```
hello ada, welcome to my library
hello aglaia, welcome to my studio
hello ada, welcome to my classroom
hello cara, welcome to my library
hello cara, welcome to my library
hello suzan, welcome to my library
```

String operations

```
In [55]: x = "XPUB says Hello! "
```

```
In [56]: x.upper()
```

```
Out[56]: 'XPUB SAYS HELLO! '
```

```
In [57]: x.title()
```

```
Out[57]: 'Xpub Says Hello! '
```

```
In [58]: x.swapcase()
```

```
Out[58]: 'xpub SAYS hELLO! '
```

```
In [59]: x.strip()
```

floats

```
In [10]: type(1.5)
```

```
Out[10]: float
```

You can turn a float into an integer, sometimes this is useful. Or the other way around.

```
In [11]: float(2)
```

```
Out[11]: 2.0
```

```
In [12]: int(1.5)
```

```
Out[12]: 1
```

strings

```
In [13]: c = "hello"
         d = "world"
```

```
In [14]: c + d
```

```
Out[14]: 'helloworld'
```

```
In [15]: type(c)
```

```
Out[15]: str
```

```
In [16]: str(2)
```

```
Out[16]: '2'
```

```
In [17]: "3" + "2"
```

```
Out[17]: '32'
```

It is possible to *access* specific characters in a string.

For example, you can select the first letter of a string with `[0]` or the last with `[-1]`.

```
In [18]: c[0]

Out[18]: 'h'

In [19]: c[-1]

Out[19]: 'o'

In [20]: c[1]

Out[20]: 'e'
```

lists

```
In [21…: xpub1 = ["log", "cara", "ada", "aglaia",
         "boyana", "suzan"]

In [22]: xpub1

Out[22]: ['log', 'cara', 'ada', 'aglaia', 'boyana',
         'suzan']

In [23]: type(xpub1)

Out[23]: list

In [24]: xpub1[0]

Out[24]: 'log'

In [25]: xpub1[-1]

Out[25]: 'suzan'
```

```
hello suzan, welcome to my studio
hello suzan, welcome to my classroom
hello suzan, welcome to my library
hello log, welcome to my studio
hello log, welcome to my classroom
hello log, welcome to my library
hello cara, welcome to my studio
hello cara, welcome to my classroom
hello cara, welcome to my library
hello boyana, welcome to my studio
hello boyana, welcome to my classroom
hello boyana, welcome to my library
hello aglaia, welcome to my studio
hello aglaia, welcome to my classroom
hello aglaia, welcome to my library
hello ada, welcome to my studio
hello ada, welcome to my classroom
hello ada, welcome to my library
```

pseudo-random numbers

You can load built-in *modules* or external *libraries* to use functionalities that other people wrote.

```
In [49]: from random import choice

In [50]: choice(xpub1)

Out[50]: 'suzan'

In [51]: choice(xpub1)

Out[51]: 'aglaia'

In [52]: choice(xpub1)

Out[52]: 'aglaia'
```

```
In [43]: s = f"hello { name }, welcome to my { item }"

In [44]: s

Out[44]: 'hello you, welcome to my house'
```

for loops

```
In [45]: for name in xpub1:
             print(name)

suzan
log
cara
boyana
aglaia
ada

In [46]: items = ["studio", "classroom", "library"]

In [47…: for name in xpub1:
             print(f"hello { name }, welcome to my {
         items[0] }")

hello suzan, welcome to my studio
hello log, welcome to my studio
hello cara, welcome to my studio
hello boyana, welcome to my studio
hello aglaia, welcome to my studio
hello ada, welcome to my studio

In [48…: for name in xpub1:
             for item in items:
                 print(f"hello { name }, welcome to my {
         item }")
```

You can also select a *range* of items in a list, which is called *slicing*:

```
In [26]: xpub1[0:3]

Out[26]: ['log', 'cara', 'ada']

In [27]: xpub1[3:]

Out[27]: ['aglaia', 'boyana', 'suzan']

In [28]: xpub1[:2]

Out[28]: ['log', 'cara']
```

You can sort a list on alphabetical order, or reversed:

```
In [29]: xpub1.sort()

In [30]: xpub1

Out[30]: ['ada', 'aglaia', 'boyana', 'cara', 'log',
         'suzan']

In [31]: xpub1.sort(reverse=True)

In [32]: xpub1

Out[32]: ['suzan', 'log', 'cara', 'boyana', 'aglaia',
         'ada']
```

string-to-list, list-to-string

string-to-list

```
In…  xpub = "XPUB focuses on the acts of making things
     public and creating publics in the age of post-
     digital networks. "
```

```
In [34]: xpub
```

```
Out[34]: 'XPUB focuses on the acts of making things
         public and creating publics in the age of
         post-digital networks. '
```

```
In [35]: xpub.split()
```

```
Out[35]: ['XPUB',
          'focuses',
          'on',
          'the',
          'acts',
          'of',
          'making',
          'things',
          'public',
          'and',
          'creating',
          'publics',
          'in',
          'the',
          'age',
          'of',
          'post-digital',
          'networks.']
```

list-to-string

```
In [36]: xpub_list = xpub.split()
```

```
In [37]: xpub_list.sort()
```

```
In [38]: xpub_list
```

```
Out[38]: ['XPUB',
          'acts',
          'age',
          'and',
          'creating',
          'focuses',
          'in',
          'making',
          'networks.',
          'of',
          'of',
          'on',
          'post-digital',
          'public',
          'publics',
          'the',
          'the',
          'things']
```

```
In [39]: " ".join(xpub_list)
```

```
Out[39]: 'XPUB acts age and creating focuses in making
         networks. of of on post-digital public
         publics the the things'
```

```
In [40]: xpub1
```

```
Out[40]: ['suzan', 'log', 'cara', 'boyana', 'aglaia',
         'ada']
```

```
In [41]: "------".join(xpub1)
```

```
Out[41]: 'suzan------log------cara------boyana------
         aglaia------ada'
```

f-strings

```
In [42]: name = "you"
         item = "house"
```