

# 0x00. Python - Variable Annotations

 Python (/python)  Background (/planning/me)



Weight: 1

Projects(/projects/current)



Project over - took place from Jul 4, 2024 6:00 AM to Jul 5, 2024 6:00 AM



An auto QA review will automatically make corrections(/to\_review)



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

**In a nutshell...**

- **Auto QA review:** 53.0/53 mandatory & 10.0/10 optional

- **Altogether: 200.0%**



Curriculums(/dashboards/my\_curriculums)

◦ Mandatory: 100.0%

◦ Optional: 100.0%

◦ Calculation:  $100.0\% + (100.0\% * 100.0\%) == 200.0\%$ 

Concepts(/concepts)

**Concepts**

Conference rooms(/dashboards/video\_rooms)



For this subject, we expect you to look at this concept:

- Advanced Python (/concepts/554)



Sandboxes(/user\_containers/current)



Tools(/dashboards/my\_tools)



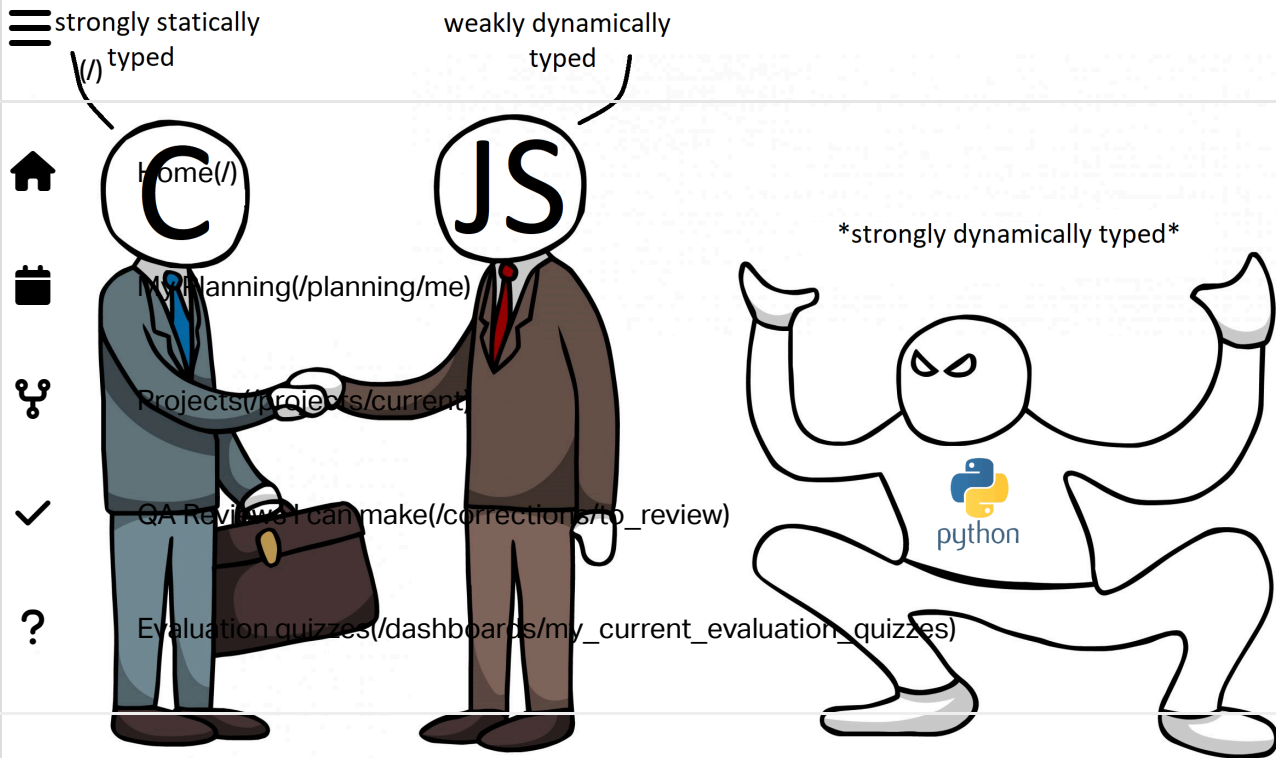
Video on demand(/dashboards/videos)



Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my\_profile)



 Curriculums(/dashboards/my\_curriculums)

## Resources

 Read or watch: Concepts(/concepts)



- Python 3 typing documentation (/rltoken/5j0OtdWh36\_HVAHKJX2gaA)
- MyPy cheat sheet (/rltoken/Fud-nrUG7x3iT6JD2Sas-g)

Conference rooms(/dashboards/video\_rooms)

## Learning Objectives

Servers(/servers)

### General



Sandboxes(/user\_containers/current)

At the end of this project, you are expected to be able to explain to anyone (/rltoken/hGUom4nCewYmroS4ii\_ZDQ), **without the help of Google:**



- Type annotations in Python 3
- How you can use type annotations to specify function signatures and variable types
- Duck typing



- Video on demand(/dashboards/videos)
- How to validate your code with mypy

## Requirements



Peers(/users/peers)

### General



- Allowed editors: `vi`, `vim`, `emacs` (`https://discord.com/app`)
- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using `python3` (version 3.7)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/env python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `pycodestyle` style (version 2.5.)
- All your files must be executable

My Profile(/users/my\_profile)





- The length of your files will be tested using `wc`
- All your modules should have a documentation ( `python3 -c (/) 'print(__import__("my_module").__doc__)'` )



- All your classes should have a documentation ( `python3 -c 'print(__import__("my_module").MyClass.__doc__)'` )



- All your functions (inside and outside a class) should have a documentation ( `python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'` )



- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)

QA Reviews I can make(/corrections/to\_review)

## Tasks



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

### 0. Basic annotations - add

mandatory



Score: 100.0% (Checks completed: 100.0%)  
Curriculums(/dashboards/my\_curriculums)

Write a type-annotated function `add` that takes a float `a` and a float `b` as arguments and returns their sum as a float



Concepts(/concepts)



#!/usr/bin/env python3  
add = \_\_import\_\_('0-add').add

Conference rooms(/dashboards/video\_rooms)



```
print(add(1.11, 2.22) == 1.11 + 2.22)
print(add.__annotations__)
```

Servers(/servers)



```
bob@dylan:~$ ./0-main.py
True
{'a': <class 'float'>, 'b': <class 'float'>, 'return': <class 'float'>}
```

Sandboxes(/user\_containers/current)



Tools(/dashboards/my\_tools)

#### Repo:



Video on demand(/dashboards/videos)

- GitHub repository: `alx-backend-python`
- Directory: `0x00-python_variable_annotations`
- File: `0-add.py`



Peers(/users/peers)

Check submission

Get a sandbox

View results



Discord(<https://discord.com/app>)



### 1. Basic annotations - concat

mandatory

My Profile(/users/my\_profile)  
Score: 100.0% (Checks completed: 100.0%)

Write a type-annotated function `concat` that takes a string `str1` and a string `str2` as arguments and returns a concatenated string

```
(/)  
bob@dylan:~$ cat 1-main.py  
#!/usr/bin/env python3  
from typing import __('1-concat').concat  
  
str1 = "egg"  
str2 = "shell"  
  
print(concat(str1, str2) == "{}{}".format(str1, str2))  
print(concat.__annotations__)  
  
bob@dylan:~$ ./1-main.py  
True  
{'str1': <class 'str'>, 'str2': <class 'str'>, 'return': <class 'str'>}
```

? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

Repo:

- GitHub repository: `alx-backend-python`
- Directory: `0x00-python-variable-annotations`
- File: `1-concat.py`

Check submission

> Get a sandbox

View results

Basic annotations (/dashboards/video\_rooms)

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a type-annotated function `floor` which takes a float `n` as argument and returns the floor of the float.

```
bob@dylan:~$ cat 2-main.py  
#!/usr/bin/env python3  
  
import math  
floor = __import__('2-floor').floor  
  
ans = floor(3.14)  
  
print(ans == math.floor(3.14))  
print(floor.__annotations__)  
print("floor(3.14) returns {}, which is a {}".format(ans, type(ans)))  
  
bob@dylan:~$ ./2-main.py  
True  
{'n': <class 'float'>, 'return': <class 'int'>}  
floor(3.14) returns 3, which is a <class 'int'>
```

My Profile(/users/my\_profile)

## Repo:



- GitHub repository: alx-backend-python
- Directory: 0x00-python\_variable\_annotations
- File: 2-floor.py



Home(/)



Check submission



Get a sandbox

View results

My Planning(/planning/me)

### 3. Basic annotations - to string

mandatory



Projects(/projects/current)

Score: 100.0% (Checks completed: 100.0%)



QA Reviews I can make(/corrections/to\_review)

Write a type-annotated function `to_str` that takes a float `n` as argument and returns the string representation of the float.



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

```
bob@dylan:~$ cat 3-main.py
```

```
#!/usr/bin/env python3
```

```
to_str = __import__('3-to_str').to_str
```



Curriculums(/dashboards/my\_curriculums)

```
pi_str = to_str(3.14)
```

```
print(pi_str == str(3.14))
```

```
print(to_str.__annotations__)
```



Concepts(/concepts)

```
print("to_str(3.14) returns {} which is a {}".format(pi_str, type(pi_str)))
```

```
bob@dylan:~$ ./3-main.py
```



Conference rooms(/dashboards/video\_rooms)

```
{'n': <class 'float'>, 'return': <class 'str'>}
```

```
to_str(3.14) returns 3.14, which is a <class 'str'>
```



Servers(/servers)

## >Repo: Sandboxes(/user\_containers/current)

- GitHub repository: alx-backend-python
- Directory: 0x00-python\_variable\_annotations
- File: 3-to\_str.py



Tools(/dashboards/my\_tools)



Video on demand(/dashboards/videos)

Check submission



Get a sandbox

View results



Peers(/users/peers)

### 4. Define variables

mandatory



Discord(<https://discord.com/app>)

Score: 100.0% (Checks completed: 100.0%)



Define and annotate the following variables with the specified values:

- `a`, an integer with a value of 1
- `pi`, a float with a value of 3.14

My Profile(/users/my\_profile)

- `i_understand_annotations` , a boolean with a value of `True`
- `school` , a string with a value of `"Holberton"`



(/)

bob@dylan: \$ cat 4-main.py

#!/usr/bin/env python3

Home(/)

`a = __import__('4-define_variables').a`

`pi = __import__('4-define_variables').pi`

`i_understand_annotations = __import__('4-define_variables').i_understand_annotations`

`school = __import__('4-define_variables').school`

Projects(/projects/current)

`print("a is a {} with a value of {}".format(type(a), a))`

`print("pi is a {} with a value of {}".format(type(pi), pi))`

`print("i_understand_annotations is a {} with a value of {}".format(type(i_understand_annotations), i_understand_annotations))`

`print("school is a {} with a value of {}".format(type(school), school))`

Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

bob@dylan:~\$ ./4-main.py

`a is a <class 'int'> with a value of 1`

`pi is a <class 'float'> with a value of 3.14`

`i_understand_annotations is a <class 'bool'> with a value of True`

`school is a <class 'str'> with a value of Holberton`



Concepts(/concepts)

Repo:

- GitHub repository: `alx-backend-python`
- Directory: `0x00-python_variable_annotations`
- File: `4-define_variables.py`



Servers(/servers)

Check submission

> Get a sandbox

View results



Sandboxes(/user\_containers/current)

## 5. Complex types - list of floats

mandatory



Tools(/dashboards/my\_tools)

Score: 100.0% (Checks completed: 100.0%)



Video on demand(/dashboards/videos)

Write a type-annotated function `sum_list` which takes a list `input_list` of floats as argument and returns their sum as a float.



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)

```
bob@dylan:~$ cat 5-main.py
#!/usr/bin/env python3
()
sum_list = __import__('5-sum_list').sum_list

Home(/)
floats = [3.14, 1.11, 2.22]
floats_sum = sum_list(floats)
print(floats_sum == sum(floats))
My Planning(/planning/me)
print(sum_list(floats) returns {} which is a {}".format(floats_sum, type(floats_sum)))
Projects(/projects/current)
bob@dylan:~$ ./5-main.py
True
QA Reviewstypingmake(typedfunctions/toreview): <class 'float'>
sum_list(floats) returns 6.470000000000001 which is a <class 'float'>
```

? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

### Repo:

- GitHub repository: alx-backend-python
- Directory: 0x00-python-variable-annotations Curriculumstypingmake(typedfunctions/toreview)
- File: 5-sum\_list.py

Concepts(/concepts)

Check submission

> Get a sandbox

View results

Conference rooms(/dashboards/video\_rooms)

mandatory

Score: 100.0% (checks completed: 100.0%)

> Write a type-annotated function `sum_mixed_list` which takes a list `mx_d_lst` of integers and floats and returns their sum as a float.

```
bob@dylan:~$ cat 6-main.py
#!/usr/bin/env python3

sum_mixed_list = __import__('6-sum_mixed_list').sum_mixed_list

print(sum_mixed_list.__annotations__)
mixed = [5, 4, 3.14, 666, 0.99]
ans = sum_mixed_list(mixed)
print(ans)
print("sum_mixed_list(mixed) returns {} which is a {}".format(ans, type(ans)))

bob@dylan:~$ ./6-main.py
typing.List[typing.Union[int, float]]
679.13
True
sum_mixed_list(mixed) returns 679.13 which is a <class 'float'>
```



Repo: My Profile(/users/my\_profile)

- GitHub repository: alx-backend-python
- Directory: 0x00-python\_variable\_annotations
- File: 6-sum\_mixed\_list.py

[Home\(/\)](#)[Check submission](#)[Get a sandbox](#)[View results](#)

## Complex types, string and int/float to tuple

[My Planning\(/planning/me\)](#)**mandatory**

Score: 100.0% (Checks completed: 100.0%)  
Projects(/projects/current)

Write a type-annotated function `to_kv` that takes a string `k` and an int OR float `v` as arguments and returns a tuple. The first element of the tuple is the string `k`. The second element is the square of the int/float `v` and should be annotated as a float.



bob@dylan:~\$ cat /tmp/evaluation\_1234567890/my\_current\_evaluation\_quizzes/  
#!/usr/bin/env python3

```
to_kv = __import__('7-to_kv').to_kv
```



Curriculums(/dashboards/my\_curriculums)

```
print(to_kv.__annotations__)
```

```
print(to_kv("eggs", 3))
```



```
print(to_kv("school", 0.02))
```

Concepts(/concepts)

```
bob@dylan:~$ ./7-main.py
```



```
{'k': <class 'str'>, 'v': typing.Union[int, float], 'return': typing.Tuple[str, f  
loat]}
```

```
('eggs', 9)
```

```
('school', 0.0004)
```



Servers(/servers)



Repo: Sandboxes(/user\_containers/current)

- GitHub repository: alx-backend-python
- Directory: 0x00-python\_variable\_annotations
- File: 7-to\_kv.py



Video on demand(/dashboards/videos)

[Check submission](#)[Get a sandbox](#)[View results](#)

Peers(/users/peers)

## 8. Complex types - functions

**mandatory**

Discord(<https://discord.com/app>)


Score: 100.0% (Checks completed: 100.0%)




Write a type-annotated function `make_multiplier` that takes a float `multiplier` as argument and returns a function that multiplies a float by `multiplier`.

[My Profile\(/users/my\\_profile\)](#)





 bob@dylan:~\$ cat 8-main.py


```
#!/usr/bin/env python3
()
make_multiplier = __import__('8-make_multiplier').make_multiplier
print(make_multiplier.__annotations__)
fun = make_multiplier(2.22)
print("{}".format(fun(2.22)))
```


 bob@dylan:~/Planning/planning/me

```
{'multiplier': <class 'float'>, 'return': typing.Callable[[float], float]}
4.9284000000000001
```


 Projects(/projects/current)

-  **Repo:** QA Reviews I can make(/corrections/to\_review)
- GitHub repository: alx-backend-python
  - Directory: 0x00-python\_variable\_annotations
  - File: 8-make\_multiplier.py

 Check submission

 > Get a sandbox

 View results


 Curriculums(/dashboards/my\_curriculums)

## 9. Let's duck type an iterable object

mandatory


 Concepts(/concepts)

Score: 100.0% (Checks completed: 100.0%)


 Conference rooms(/dashboards/video\_rooms)

Annotate the below function's parameters and return values with the appropriate types

```
def element_length(lst):
    Servers(/servers)
    return [(i, len(i)) for i in lst]
```


 bob@dylan:~/Sandboxes/user\_containers/current

```
#!/usr/bin/env python3
```

 element\_length(/dashboards/my\_tools)


```
9-element_length').element_length
```

```
print(element_length.__annotations__)
```

 Video on demand(/dashboards/videos)

bob@dylan:~\$ ./9-main.py

```
{'lst': typing.Iterable[typing.Sequence], 'return': typing.List[typing.Tuple[typing.Sequence, int]]}
```

 Peers(/users/peers)

## Repo:

- Discord(<https://discord.com/app>)
- GitHub repository: alx-backend-python
- Directory: 0x00-python\_variable\_annotations
- File: 9-element\_length.py



 Check submission

 > Get a sandbox

 View results

## 10. Duck typing - first element of a sequence

#advanced

(/)

Score: 100.0% (Checks completed: 100.0%)



Home(/)

Augment the following code with the correct duck-typed annotations:



# The types of the elements of the input are not known

My Planning(/planning/me)

```
def safe_first_element(lst):
    if lst:
        return lst[0]
    else:
        return None
```



Projects(/projects/current)



QA Reviews I can make(/corrections/to\_review)

bob@dylan:~\$ cat 100-main.py

#!/usr/bin/env python3



Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

```
safe_first_element = __import__('100-safe_first_element').safe_first_element
```

```
print(safe_first_element.__annotations__)
```



Curriculums(/dashboards/my\_curriculums)

bob@dylan:~\$ ./100-main.py

```
{'lst': typing.Sequence[typing.Any], 'return': typing.Union[typing.Any, NoneType]}
```



Concepts(/concepts)



Repo: Conference rooms(/dashboards/video\_rooms)

- GitHub repository: alx-backend-python



- Directory: 0x00-python\_variable\_annotations

Servers(/servers)

- File: 100-safe\_first\_element.py



Sandboxes(/user\_containers/current)

Check submission

Get a sandbox

View results



Tools(/dashboards/my\_tools)



## 11. More involved type annotations

Video on demand(/dashboards/videos)

#advanced

Score: 100.0% (Checks completed: 100.0%)



Peers(/users/peers)

Given the parameters and the return values, add type annotations to the function

Hint: look into TypeVar



Discord(<https://discord.com/app>)

```
def safely_get_value(dct, key, default = None):
    if key in dct:
        return dct[key]
    else:
        return default
```

My Profile(/users/my\_profile)



```

bob@dylan:~$ cat 101-main.py
#!/usr/bin/env python3
()
safely_get_value = __import__('101-safely_get_value').safely_get_value
annotations = safely_get_value.__annotations__
Home()
print("Here's what the mappings should look like")
for k, v in annotations.items():
    print(f"My Planning(/planning/{k})")

bob@dylan:~$ ./101-main.py
Here's what the mappings should look like
dct: typing.Mapping
key: typing.Any
default: typing.Union[typing.Any, Callable[[typing.Any, typing.Any], typing.Any]]
return: typing.Union[typing.Any, ~T]

```

? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

## Repo:

- GitHub repository: alx-backend-python
- Directory: 0x00-python-variable-annotations
- File: 101-safely\_get\_value.py

Concepts(/concepts)

Check submission

> Get a sandbox

View results

Conference rooms(/dashboards/video\_rooms)

#advanced

Score: 100.0% (checks completed: 100.0%)

Use mypy to validate the following piece of code and apply any necessary changes.

```

def zoom_array(lst: Tuple, factor: int = 2) -> Tuple:
    zoomed_in: Tuple = [
        item for item in lst
        for i in range(factor)
    ]
    return zoomed_in

```

```

array = [12, 72, 91]
Peers(/users/peers)
zoom_2x = zoom_array(array)

```

```

zoom_3x = zoom_array(array, 3)
Discord(https://discord.com/app)

```



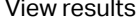
My Profile(/users/my\_profile)


```
bob@dylan:~$ mypy 102-type_checking.py
Success: no issues found in 1 source file
bob@dylan:~$ cat 102-main.py
#!/usr/bin/env python3

__import__('102-type_checking').zoom_array


print(zoom_array.__annotations__)
My Planning(/planning/me)
bob@dylan:~$ ./102-main.py
{'lst': typing.Tuple, 'factor': <class 'int'>, 'return': typing.List}
Projects(/projects/current)
```

- ✓ Repo: QA Reviews I can make(/corrections/to\_review)
- GitHub repository: alx-backend-python
  - Directory: 0x00-python\_variable\_annotations
  - File: 102-type\_checking.py
- ? Evaluation quizzes(/dashboards/my\_current\_evaluation\_quizzes)

 Check submission  Get a sandbox  View results


 Curriculums(/dashboards/my\_curriculums)

 Concepts(/concepts)

 Conference rooms(/dashboards/video\_rooms)


Copyright © 2024 ALX, All rights reserved.

 Servers(/servers)

 Sandboxes(/user\_containers/current)

 Tools(/dashboards/my\_tools)

 Video on demand(/dashboards/videos)

 Peers(/users/peers)

 Discord(<https://discord.com/app>)



My Profile(/users/my\_profile)