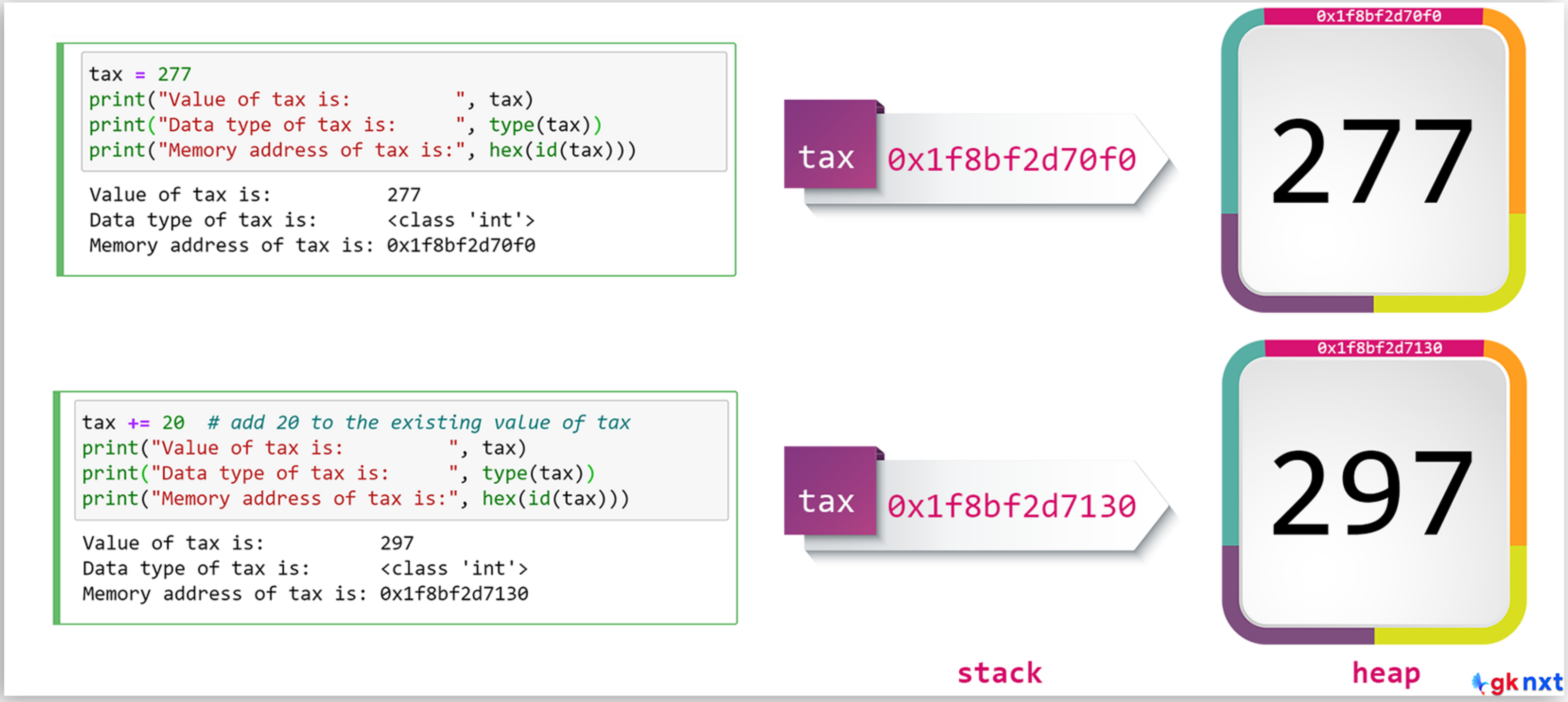
Python Bootcamp & Masterclass

integers

gknxt

An integer is a whole number with no fractional component (with no decimal places) For example, `-66`, `0`, `247` are all integers. Integers are immutable (they can't be modified)



```
tax = 277
print("Value of tax is:          ", tax)
print("Data type of tax is:      ", type(tax))
print("Memory address of tax is:", hex(id(tax)))
```

```
Value of tax is:          277
Data type of tax is:      <class 'int'>
Memory address of tax is: 0x1f8bf2d70f0
```

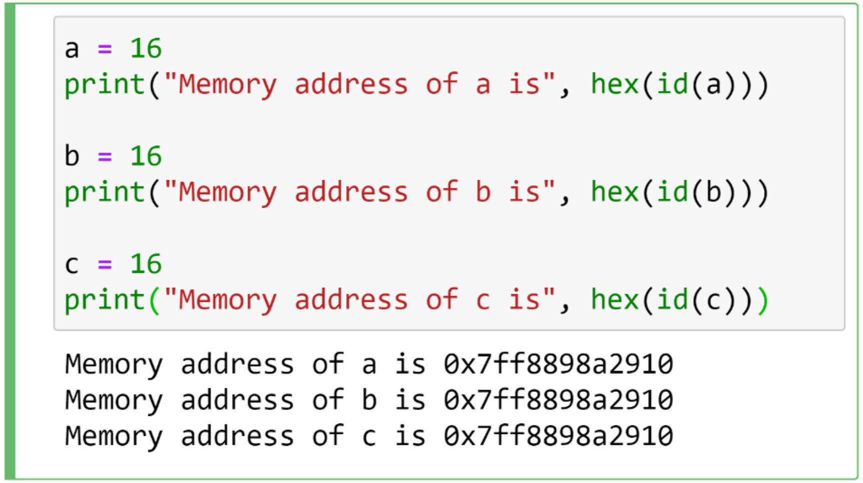
```
tax += 20  # add 20 to the existing value of tax
print("Value of tax is:          ", tax)
print("Data type of tax is:      ", type(tax))
print("Memory address of tax is:", hex(id(tax)))
```

```
Value of tax is:          297
Data type of tax is:      <class 'int'>
Memory address of tax is: 0x1f8bf2d7130
```

gknxt

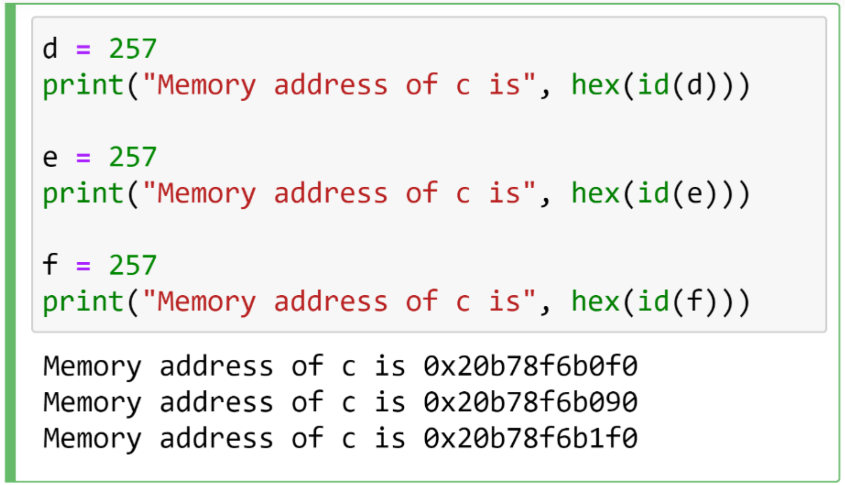
integer interning

All objects of integer type having values from -5 to 256 are unique and are shared for execution efficiency (i.e. if many integer objects are created to store 16, only one object will be used and all variables of value 16 point to that object)



```
a = 16
print("Memory address of a is", hex(id(a)))

b = 16
print("Memory address of b is", hex(id(b)))

c = 16
print("Memory address of c is", hex(id(c)))
```

```
Memory address of a is 0x7ff8898a2910
Memory address of b is 0x7ff8898a2910
Memory address of c is 0x7ff8898a2910
```



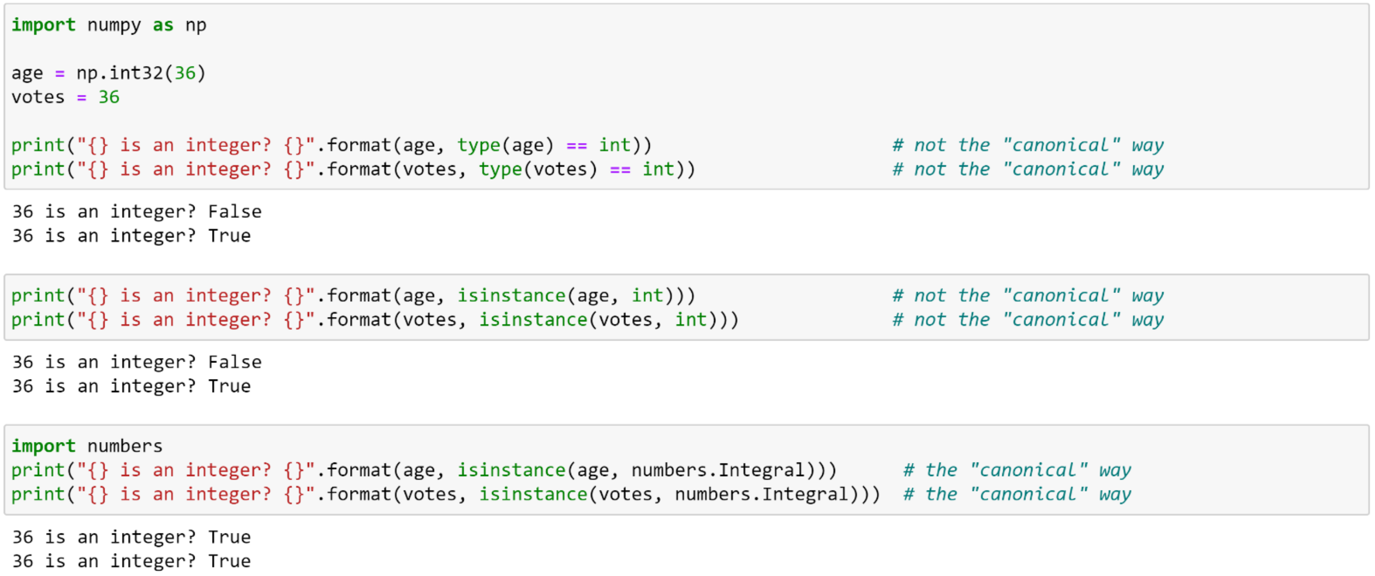
```
d = 257
print("Memory address of c is", hex(id(d)))

e = 257
print("Memory address of c is", hex(id(e)))

f = 257
print("Memory address of c is", hex(id(f)))
```

```
Memory address of c is 0x20b78f6b0f0
Memory address of c is 0x20b78f6b090
Memory address of c is 0x20b78f6b1f0
```

gknxt

Is it an integer?

If an object needs to be checked if it is an integer object, isinstance( ) method can be used. It is preferred over the type( ) method. The best practice is to compare against number.Integral as it can work even with numpy integers.



```
import numpy as np

age = np.int32(36)
votes = 36

print("{} is an integer? {}".format(age, type(age) == int))                 # not the "canonical" way
print("{} is an integer? {}".format(votes, type(votes) == int))             # not the "canonical" way
```

```
36 is an integer? False
36 is an integer? True
```

```
print("{} is an integer? {}".format(age, isinstance(age, int)))             # not the "canonical" way
print("{} is an integer? {}".format(votes, isinstance(votes, int)))         # not the "canonical" way
```

```
36 is an integer? False
36 is an integer? True
```

```
import numbers
print("{} is an integer? {}".format(age, isinstance(age, numbers.Integral)))      # the "canonical" way
print("{} is an integer? {}".format(votes, isinstance(votes, numbers.Integral)))  # the "canonical" way
```

```
36 is an integer? True
36 is an integer? True
```

gknxt

No size limit for int

There’s no size limit for an integer (limited only by system memory) The amount of memory/storage needed for an integer (as per the CPython implementation) depends on the sign/magnitude of that integer, and the python version/OS/hardware used

```python
# There’s no limit to how large an integer can be
z = 9223372036854775808234343434343433232323232323232323232399999999937283723245
z
```

```
9223372036854775808234343434343433232323232323232323232399999999937283723245
```

```python
import sys
print("Memory used for 0 (zero): {0} {1}" .format(sys.getsizeof(0), 'bytes'))
print("Memory used  for 1 (one): {0} {1}" .format(sys.getsizeof(1), 'bytes'))
print("Memory used  for 2 ** 30: {0} {1}" .format(sys.getsizeof(2 ** 30), 'bytes'))
print("Memory used  for 2 ** 60: {0} {1}" .format(sys.getsizeof(2 ** 60), 'bytes'))
print("Memory used  for 2 ** 90: {0} {1}" .format(sys.getsizeof(2 ** 90), 'bytes'))
print("Memory used  for 2 **120: {0} {1}" .format(sys.getsizeof(2 ** 120), 'bytes'))
```

```
Memory used for 0 (zero): 24 bytes
Memory used  for 1 (one): 28 bytes
Memory used  for 2 ** 30: 32 bytes
Memory used  for 2 ** 60: 36 bytes
Memory used  for 2 ** 90: 40 bytes
Memory used  for 2 **120: 44 bytes
```

digit separator

- Commas or spaces as separators between the digits of an integer are not allowed

- Underscores can be used if separation between the digits is needed for readability

```
# Commas or spaces as separators between the digits of an integer are not allowed.
pop = 927 284 000
pop = 927,284,000
```

```
  File "<ipython-input-13-6f90fc55ccd1>", line 2
    pop = 927 284 000
              ^
SyntaxError: invalid syntax
```

```
# Underscores can be used if separation between the digits is needed for readability
pop = 927_284_000
pop
```

```
927284000
```

gknxt

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt

Python Bootcamp & Masterclass

Thank You

for your Rating & Review

gknxt