Built-In Primitive Types in Python

- Numbers
- Booleans
- Strings ## Variables

In [1]:

```
students_count = 1000 # int
rating = 4.99  # float
is_published = False # boolean
course_name = "Python Programming" # string
print(students_count, rating, is_published, course_name)
```

```
1000 4.99 False Python Programming
```

Variable Names

- Descriptive and meaningful
- Lower case letters to name variables
- Underscore to separate multiple words

Strings

In [2]:

```
course = "Python Programming"
message = """

Hi John,

This is Mosh from codewithmosh.com.

"""
print(course, message)
```

```
Python Programming

Hi John,

This is Mosh from codewithmosh.com.
```

- Length
- Indexing

In [3]:

```
print(len(course))
```

```
18
```

In [4]:

```
course[0], course[-1], course[0:3], course[0:], course[:-1], course[:]
```

Out[4]:

```
('P',
 'g',
 'Pyt',
 'Python Programming',
 'Python Programmin',
 'Python Programming')
```

Escape Sequences

What if we want to add a " in a string? We use the escape character \ (back slash).
\" is an escape sequence.

- \'
- \\
- \n

In [5]:
```
course = "Python \"Programming"
print(course)
course = "Python \\Programming"
print(course)
course = "Python \nProgramming"
print(course)
```

```
Python "Programming
Python \Programming
Python
Programming
```

Formatted Strings

In [6]:
```
first = "Mosh"
last = "Hamedani"
full = first + " " + last
print(full)
```

```
Mosh Hamedani
```

The above is not neat.

In [7]:
```
first = "Mosh"
last = "Hamedani"
full = f"{first} {last}"
print(full)
```

```
Mosh Hamedani
```

We can put any kind of expression in between curly braces.

In [8]:
```
print(f"{len(first)} {2 + 2}")
```

```
4 4
```

String Methods

Everything in Python is an object, which has functions we call methods that we can access using the dot notation.

Upper/lower case

In [9]:
```
course = "  python Programming  "
```

```
print(course.upper())
print(course.lower())
print(course.title())
print(course)
```

```
  PYTHON PROGRAMMING
  python programming
  Python Programming
  python Programming
```

Strip

In [10]:

```
print(course.strip()) # remove the white space from both the beginning and end of a st
print(course.lstrip())
print(course.rstrip())
print(course)
```

```
python Programming
python Programming
  python Programming
  python Programming
```

Find

In [11]:

```
print(course.find("Pro")) # return the index of what we want
print(course.find("pro")) # -1 means failure to find it
```

```
9
-1
```

Replace

In [12]:

```
print(course.replace("p", "j"))
```

```
  jython Programming
```

In

In [13]:

```
print("pro" in course)
print("Pro" in course)
print("swift" not in course)
```

```
False
True
True
```

Numbers

- Interger
- Float
- Complex numbers

In [14]:

```
x = 1 + 2j # complex number
print(x)
```

```
(1+2j)
```

Standard arithmetic operators

In [15]:

```
print(10 + 3) # addition
print(10 - 3) # subtraction
print(10 * 3) # multiplication
print(10 / 3) # division
print(10 // 3) # exact division
print(10 % 3) # modulus
print(10 ** 3) # exponent
```

```
13
7
30
3.3333333333333335
3
1
1000
```

Augmented assignment operator

In [16]:

```
x = 10
x = x + 3
x += 3 # also for -, *, / ......
print(x)
```

```
16
```

Work with Numbers

In [17]:

```
round(2.9)
```

Out[17]:

```
3
```

In [18]:

```
abs(-1.9)
```

Out[18]:

```
1.9
```

Python 3 Math Module
https://docs.python.org/3/library/math.html

In [19]:

```
import math
math.ceil(2.2)
```

Out[19]:

```
3
```

In [20]:

```
math.cos(-math.pi)
```

Out[20]:

```
-1.0
```

Type Conversion

In [21]:

```
x = input("x: ")
y = x + 1
```

```
---------------------------------------------------------------------------
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-21-b7bcf0d05f4c> in <module>
      1 x = input("x: ")
----> 2 y = x + 1

TypeError: can only concatenate str (not "int") to str
```

What value did we type in?

In [22]:
```
x
```

Out[22]:
```
'18'
```

In [23]:
```
print(int(x), float(x), bool(x), str(x))
```

```
18 18.0 True 18
```

Boolean falsy

- ""
- 0
- None

Anything else would be *True*.

In [24]:
```
bool("")
```

Out[24]:
```
False
```

In [25]:
```
bool(" ")
```

Out[25]:
```
True
```

In [26]:
```
bool(0)
```

Out[26]:
```
False
```

In [27]:
```
bool(3)
```

Out[27]:
```
True
```

In [28]:
```
bool(None)
```

Out[28]:
```
False
```

In [29]:
```
bool("False")
```

Out[29]:
```
True
```