Python (W3 schools)

> python -- version

> python filename.py.

```
print("Hello world!")
```

python command line -

> python (or) > py.

... python commands ...

> exit()

* Indentation.
* Variables. - x = 5 - case sensitive.
* Comments - #
  multi line - """ - triple double-quotes.

* Casting - to specify data type of var.

* Get type → type(x)

* x = "John" is same as x = 'John'
  quotes for string.

* Variable names.
  start w/ letter or underscore (_)
  can't start w/ number.
  A-z, 0-9., underscore.
  Case sensitive.

* x, y, z = "Ap", "Or", "Ch"
  x = y = z = "Ap"

* Unpacking - extract coll^n values into vars.

  fruits = ["ap", "or", "ch"]

  x, y, z = fruits.

* Output

```
print(x)
print(x, y, z)
print(x + y + z)
```

* global variables.

global vs local scope

global keyword. - to change global var inside fn.

Built-in data types -

Text - str
Numeric - int, float, complex
Sequence - list, tuple, range
Mapping - dict
set - set, frozenset
boolean - bool
binary - bytes, bytearray, memoryview.

type(x)

x = "Hello world"	str
x = 20	int
x = 20.5	float
x = 1j	complex.
x = ['apple', 'banana', 'cherry']	list
x = ('apple', 'banana', 'cherry')	tuple
x = range(5)	range
x = {'name': 'john', 'age': 35}	dict
x = {'apple', 'banana', 'cherry'}	set
x = frozenset({'apple', 'banana', 'cherry'})	
x = True	boolean
x = b'Hello'	bytes
x = bytearray(5)	bytearray.
x = memoryview(bytearray(5))	

Int - whole num
+ve or -ve
unlimited length.

Float - +ve or -ve
one or more decimals.
scientific nums with 'e'

type conversion.

Random Num. -
import random library
print(random.randrange(1,10))

Multi line strings - ''' or """

Strings are arrays - arrays of bytes.
representing unicode characters.

a = "banana" a.xxx()

len(a) - length of string

slicing a[2:5]
slice from the start
slice to the end.
negative indexing.
upper case - a.upper()
lower.
strip - trim whitespaces beg/end.
replace.
split. → to list
concat - using +
format.

escape char - \ — backslash

All string methods return new values. They do not change the original string.

capitalize()

casefold()

casefold vs lower.

↓ unicode text / user input.

↳ purely ASCII text

casefold() is a more aggressive version of lower() that is set up to make unique unicode characters more comparable.

lower() will require less memory or less time bcz there are no lookups and only deals w/ 26 cc's.

~~Ans~~ ASCII std - 256 characters

Unicode std - 143859 characters.

center ( )	find ( )	is decimal ( )
count ( )	format ( )	is digit ( )
encode ( )	format_map ( )	is identifier ( )
endswith ( )	index ( )	is lower ( )
expandtabs ( )	is alnum ( )	is numeric ( )

isprintable ( )	ljust ( )	replace ( )
is space ( )	lower ( )	rfind ( )
istitle ( )	lstrip ( )	rindex ( )
is upper ( )	maketrans ( )	rjust ( )
join ( )	partition ( )	rpartition ( )

rsplit ( )	startswith ( )	translate ( )
rstrip ( )	strip ( )	upper ( )
split ( )	swapcase ( )	zfill ( )
splitlines ( )	title ( )	

bool()

False - (), [], {}, "", 0, None

isinstance(x, int).

Operators

+ - * / % ** //

↓ exponent ↓ floor division

logical - and or not

identity - is, is not

compare objs, not if they are equal, but if they are actually the same obj, with the same memory location.

Membership - in, not in

Bitwise - AND, OR, XOR, NOT

List - indexed, ordered, mutable, allow duplicates.

list() → constructor.

- .insert()
- .append()
- .extend()
- .remove() item
- .pop() idx

  del
- .clear()

List Comprehension.

newlist = [expression for item in iterable if condition == True]

Sort Lists.

```
sort()
sort(reverse = True)
sort(key = myfunc)
sort(key = str.lower)
```

Copy

- copy()

```
list(1)
```

Concatenate → +

- extend()

Tuples ( )

ordered, unchangeable, allow dups.

("apple",) → Tuple w/ single item needs comma.

len ()

type ()

tuple ()

	ordered	changeable.	Duplicates.
List	✓	✓	✓
Tuple.	✓	✗	✓
Dictionary	✓	✓	✗
Set	✗	✗	✗

'in' keyword.

add tuple to tuple → +

del tuple

packing & unpacking.

unpacking with *

Join → +

multiply → *

count ('x') , index ('x')

Sets - { }

can't use index or key to access items.

add ( ) → add new items.
update ( ) → add iterable into set.
remove ( )
discard ( )
pop ( ) → remove last item, whichever
clear ( ) →
del. < >
union ( ) → exclude dups.
intersection_update ( ) →
intersection ( )
symmetric_difference_update ( ) →
symmetric_difference ( )

Dictionaries.

len ( )

type ( )

get (key)

keys ( )

values ( )

items ( )

update ({key : values})

pop (key)

popitem ( ) — last inserted item

del < >

clear ( )

loop.

```
for k, v in this.items ( )
    print (k, v)
```

copy ( ).

dict ( )

set default ()

if elif else.

'pass' statement -

while, break, continue, else

for, break, continue, range(), else

```
def my_func():
    print().

my_func()
```

Arbitrary arguments - *args.
**kwargs

Lambda fn -

lambda arguments : expression