Syntax תחביר

העקרונות

- מה לא? {} () ;
- מה כן? : tab
- לא פייתון!

```
if (brackets) {
if( !tab || ! any(“:”)) {return !python;}
else {return “just bad python”;} }
```

- כן פייתון!

```
if not brackets and tab:
        return python
```

שבירת שורות וטאבים

- שבירה מפורשת \
- שבירה בתוך מערך ,
- כמה הצהרות בשורה אחת ; אבל זה מכוער
- להזכירכם במקום זה אפשר x,y=1,2

הזחה זה כל הסיפור

בלוק ראשי

בלוק משני 1

בלוק 3

בתוך בלוק 3

בלוק משני 2

בתוך בלוק משני 2

נסקור לרגע את כל מילות המפתח

- import keyword
- print(keyword.kwlist)

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Conditions תנאים

- if condition:
  do

- if condition:
  do
  else:
  do_2

- if condition:
  do
  elif condition_2:
  do_2
  else:
  do_3

- if condition:
  do
  if condition_2:
  do_2
  else:
  do_3
  else:
  do_4

לולאות Loops

```
while condition:
        do_something
```

- אבל צריך לשבור את זה
- לולאות for בפייתון מאוד שונות משפות אחרות וצריך לדעת איך הכי טוב לעבוד איתן

```
for item in items:
        do(item)
```

- בניגוד ל-

```
for (i=0; i<items.length; i++){
        do(items[i]);
}
```

- מה עושים אם רוצים מונה?

שיטות עבודה עם for בפייתון

- יש כאלו שנוטים לבנות מונה "מלאכותי" שדומה לשפות אחרות

```
k=0
while k<len(items):
        do(items[k])
        k += 1
OR
k=0
for item in items:
        do(item,k)
        k +=1
```

- לא מומלץ!
- מומלץ להתרגל לרעיון הזה שאנחנו לא מחויבים לאנדקס כל הזמן. ובכל זאת...

לולאות for עם index

- השיטה הכי פייתונית ושכיחה ללולאות עם מספר מוגדר של צעדים היא

```
for i in range(n):
    f(i)
```

- איך כותבים range? דומה לאינדקסים range(start,end,step)
- נכתוב לולאה של המספרים הזוגיים בין 4 ל-50
- יש מצבים שרוצים להשתמש בזה, כשיש סיבה מיוחדת להשתמש. זוכרים?

```
for indx, item in enumerator(items):
    do_something_important_with_index(item,indx)
```

עוד דוגמאות for יחד עם if

```
for item in items:
        if item==“stop”:
                break

for i in range(20):
        if i==5:
                continue
        elif i==19:
                print(“almost done”)
```

מה חסר לדעתכם בקוד הזה? (בעיה בתפקוד לא שגיאה)

in

- במילת המפתח in משתמשים גם עם for וגם עם if

```
>>> for item in items:
...     do(item)
>>> if item in items:
...     print(items)
>>> if 5 in [1,6,7,5]:
```

פונקציות

- def my_function():
  print(“inside function”)
- def my_arg_function(my_arg):
  print(my_arg)
- def my_return_function():
  return 1
- def my_predefined_arg(my_arg=5):
  print(my_arg)

function args, *args, ארגומנטים לפונקציה **kwargs

```
def add_2_args(x,y):
        return x+y
def add_n_args(nums):
        let’s write it on the whiteboard
add_n_args([3,6,2])
add_n_args(3,6,2) ???
def add_n_args(*nums):
Attention! Try nums[1] = 8
```

**kwargs, unpack

```
def keywords(**kwargs):
keywords(a="value", b="another value")
kwargs → keys, kwargs.values() → values
def order(x,y,*args, **kwargs):


pack = [ 1,4,6]
add_n_args(*pack)
```

Iterate on kwargs

```
for key, value in kwargs.items():
        print(key + ": " + value)
```