El futuro digital es de todos MinTIC

Week 3 Class 1

Main Course

Student Worksheet

UNIVERSIDAD SERGIO ARBOLEDA

MisiónTIC2022

Understanding Scanning

Part 1. **Read the following information. Have you ever used scanning before?**

Scanning is a strategy to read a text quickly in order to find specific information.

Example: you need to know the specific number of people that are using Python around the world… So, when you scan the text, you have this question in your mind and you read the passage only to find the answer.

Follow these steps to scan any text:

1. What's your question? Make sure **you know what you are looking for**
2. Look through the text to find specific info: **keywords, numbers and phrases**
3. Scan the **title, subtitles and abstract** to see **relevant content**

Part 2. **Watch the video of the activity, take notes of the important information and answer the following questions:**

https://www.youtube.com/watch?v=LbIsVPShyyU&ab_channel=Takayuuuki

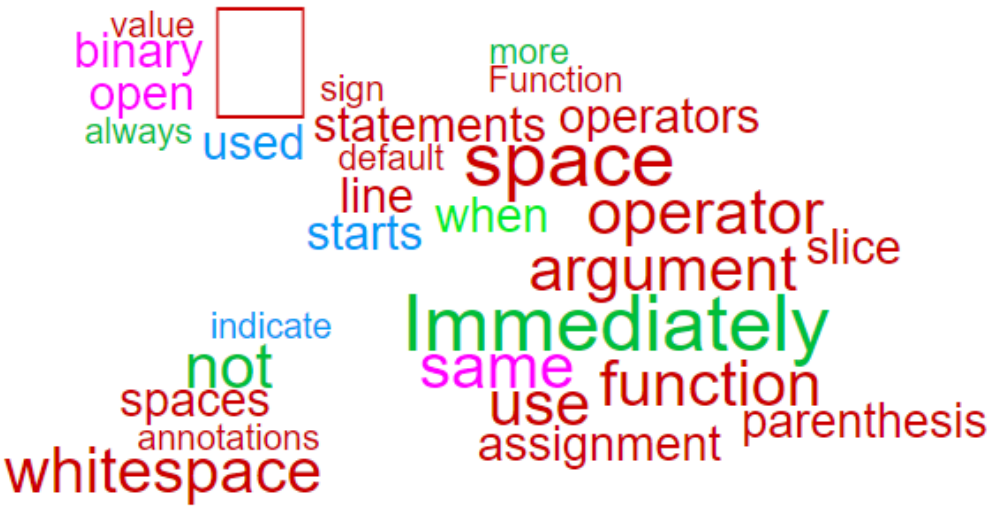
1. Why is scanning useful?
2. Do you have to read the full text?
3. Write down three conclusions about scanning.

.

I Before you read

A. Predicting

Look at the Word Cloud of the most frequent words in the text you are going to read. What might be the main topic of the text?



B. Keywords

1. **Which words you do not know and have to look up in a dictionary?**
   Check the dictionary and write down the words you do not know.

2. **Check the pronunciation of the difficult words on the dictionary and practice.**

C. Skimming

1. Skim the text and match the headings with the sections.

Section 1	_____	a. Whitespace in Expressions and Statements
Section 2	_____	b. Other Recommendations

II. While you read

1. Scan the text and answer the following questions.

1. What's the amount of whitespace that both colons must have?

2. What's the amount of whitespace that you can use on both sides of a binary operator?

2. Skim the text and identify the symbols, then write the correct option in each case.

	Answer	False
1. Parentheses		a. ,
2. Brackets		b. ;
3. Braces		c. :
4. Comma		d. {}
5. Semicolon		e. ()

6. Colon		f. [ ]
7. Pointy bracket or inequality sign		g. < >

3. **You are going to read an article about coding in Python. Before you read, decide whether the statements below are 'T' (true) or 'F' (false). Then read the text on the next page to confirm or correct your answers.**

	True	False
1. White space isn't relevant		
2. White space is easy to identify		
3. Function annotations should always have spaces around		
4. Always use white space for multi-clause statements		

PEP* 8 -- Style Guide for Python Code

Section 1. _____

Pet Peeves
Avoid extraneous whitespace in the following situations:

- Immediately inside parentheses, brackets or braces:

```
# Correct:

spam(ham[1], {eggs: 2})

# Wrong:

spam( ham[ 1 ], { eggs: 2 } )
```

- Between a trailing comma and a following close parenthesis:

```
# Correct:

foo = (0,)

# Wrong:

bar = (0, )
```

- Immediately before a comma, semicolon, or colon:

```
# Correct:

if x == 4: print x, y; x, y = y, x

# Wrong:
```

```
if x == 4 : print x , y ; x , y = y , x
```

- However, in a slice the colon acts like a binary operator, and should have equal amounts on either side (treating it as the operator with the lowest priority). In an extended slice, both colons must have the same amount of spacing applied. Exception: when a slice parameter is omitted, the space is omitted:

```
# Correct:

ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]

ham[lower:upper], ham[lower:upper:], ham[lower::step]

ham[lower+offset : upper+offset]

ham[: upper_fn(x) : step_fn(x)], ham[:: step_fn(x)]

ham[lower + offset : upper + offset]

# Wrong:

ham[lower + offset:upper + offset]

ham[1: 9], ham[1 :9], ham[1:9 :3]

ham[lower : : upper]

ham[ : upper]
```

- Immediately before the open parenthesis that starts the argument list of a function call:

```
# Correct:

ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]

ham[lower:upper], ham[lower:upper:], ham[lower::step]
```

```
ham[lower+offset : upper+offset]

ham[: upper_fn(x) : step_fn(x)], ham[:: step_fn(x)]

ham[lower + offset : upper + offset]

# Wrong:

ham[lower + offset:upper + offset]

ham[1: 9], ham[1 :9], ham[1:9 :3]

ham[lower : : upper]

ham[ : upper]
```

- Immediately before the open parenthesis that starts the argument list of a function call:

```
# Correct:
spam(1)
# Wrong:
spam (1)
```

- Immediately before the open parenthesis that starts an indexing or slicing:

```
# Correct:

dct['key'] = lst[index]

# Wrong:

dct ['key'] = lst [index]
```

- More than one space around an assignment (or other) operator to align it with another:

```
# Correct:

x = 1

y = 2

long_variable = 3

# Wrong:

x         = 1

y         = 2

long_variable = 3
```

Section 2. _____

Avoid trailing whitespace anywhere. Because it's usually invisible, it can be confusing: e.g. a backslash followed by a space and a newline does not count as a line continuation marker. Some editors don't preserve it and many projects (like CPython itself) have pre-commit hooks that reject it.
Always surround these binary operators with a single space on either side: assignment (=), augmented assignment (+=, -= etc.), comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not), Booleans (and, or, not).
If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies). Use your own judgment; however, never use more than one space, and always have the same amount of whitespace on both sides of a binary operator:

```
# Correct:

i = i + 1
```

```
submitted += 1

x = x*2 - 1

hypot2 = x*x + y*y

c = (a+b) * (a-b)

# Wrong:

i=i+1

submitted +=1

x = x * 2 - 1

hypot2 = x * x + y * y

c = (a + b) * (a - b)
```

- Function annotations should use the normal rules for colons and always have spaces around the -> arrow if present. (See Function Annotations below for more about function annotations.):

```
:# Correct:

def munge(input: AnyStr): ...

def munge() -> PosInt: ...

# Wrong:

def munge(input:AnyStr): ...

def munge()->PosInt: ...
```

- Don't use spaces around the = sign when used to indicate a keyword argument, or when used to indicate a default value for an unannotated function parameter:

```
# Correct:

def complex(real, imag=0.0):

    return magic(r=real, i=imag)

# Wrong:

def complex(real, imag = 0.0):

    return magic(r = real, i = imag)
```

- When combining an argument annotation with a default value, however, do use spaces around the = sign:

```
# Correct:

def munge(sep: AnyStr = None): ...

def munge(input: AnyStr, sep: AnyStr = None, limit=1000): ...

# Wrong:

def munge(input: AnyStr=None): ...

def munge(input: AnyStr, limit = 1000): ...

# Wrong:

def munge(input: AnyStr=None): ...

def munge(input: AnyStr, limit = 1000): ...
```

- Compound statements (multiple statements on the same line) are generally discouraged:

```
# Correct:

if foo == 'blah':
```

```
    do_blah_thing()

do_one()

do_two()

do_three()
```

- # Wrong:
- if foo == 'blah': do_blah_thing()
- do_one(); do_two(); do_three()

- While sometimes it's okay to put an if/for/while with a small body on the same line, never do this for multi-clause statements. Also avoid folding such long lines!

Rather not:

```
# Wrong:

if foo == 'blah': do_blah_thing()

for x in lst: total += x

while t < 10: t = delay()
```

Definitely not:

```
# Wrong:

if foo == 'blah': do_blah_thing()

else: do_non_blah_thing()
```

```
try: something()

finally: cleanup()



do_one(); do_two(); do_three(long, argument,

                        list, like, this)



if foo == 'blah': one (); two(); three()
```

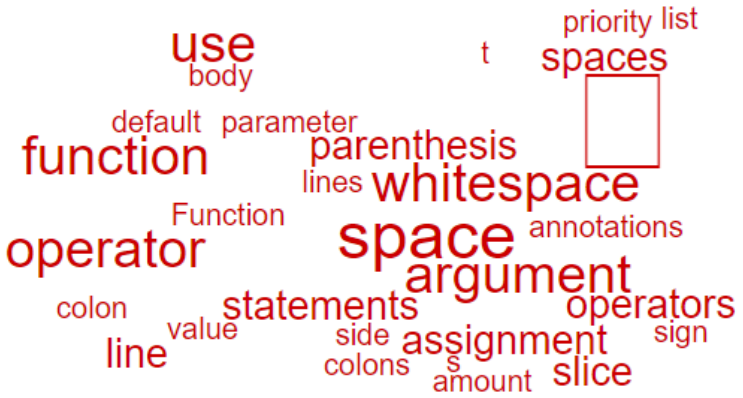
Click on the link below to read the full text: https://www.python.org/dev/peps/pep-0008/

III After you read

1. Vocabulary

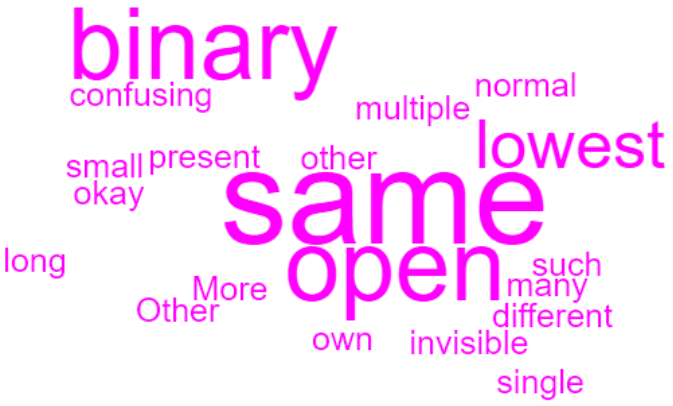
The Word Cloud below show the most frequent verbs in the text. Find them and highlight them in the text. Do you know what they mean? **If not, look them up in a dictionary.**

https://www.wordreference.com/



1. Common Adjectives in Python language

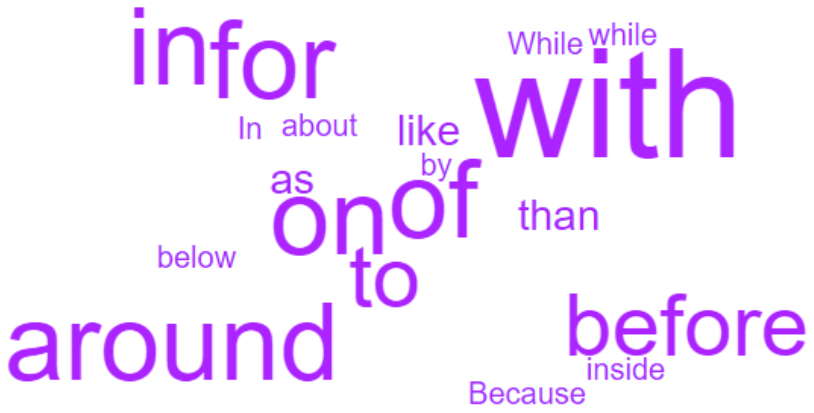
In this text, you can find these adjectives. Look at the word cloud below. **Do you know all these adjectives? Choose 10 adjectives and write in the front the opposite.**



Adjective	Antonym
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

2. Prepositions and connectors

Prepositions in the Word Cloud below are used in the text. Find them and organize them in the table below based on what they express in the text. You can also add your own category.



Time	While, Before
Place	Around, on, in, to, inside
Sequence	Before
Company	With, for, to
Comparison	Like, than, same
Add your category Reason	
Add your category	

Understanding Scanning

Part 1. **Read the following statements and mark Yes or No taking into account your learning process:**

Statements	Yes	No
1. I can explain to my partners what scanning is		
2 I can scan a text to find specific information		
3. I can manage unknown vocabulary		
4. I can use scanning and other strategies to get some specific information from a text.		

Part 2. **Write down 2 aspects you learned about Python and 1 tip you can recommend a friend for checking specific information in a text:**

Answer Key

I. Before you read

c. Skimming

Skim the text and match the headings with the sections.

Section 1	____a_____	a. Whitespace in Expressions and Statements
Section 2	___b_____	b. Other Recommendations

II. While you read

1. Scan the text and answer the following questions.

1. What's the amount of whitespace that both colons must have?

The same amount of spacing applied.

2. What's the amount of whitespace that you can use on both sides of a binary operator?

The same amount of whitespace on both sides of a binary operator

2. Skim the text and identify the symbols, then write the correct option in each case.

	Answer	False
1. Parentheses	e	a. ,
2. Brackets	f	b. ;

3. Braces	d	c. :
4. Comma	a	d. {}
5. Semicolon	b	e. ()
6. Colon	c	f. [ ]
7. Pointy bracket or inequality sign	g	g. < >

2. **You are going to read an article about coding in Python. Before you read, decide whether the statements below are 'T' (true) or 'F' (false). Then read the text on the next page to confirm or correct your answers.**

	True	False
1. White space isn't relevant		X
2. White space is easy to identify		X
3. Function annotations should always have spaces around	X	
4. Always do this for multi-clause statements		X

III. After you read

3. Prepositions and connectors

Time	While, Before
Place	

	Around, on, in, to, inside
Sequence	Before
Company	With, for, to
Comparison	Like, than, same
Add your category Reason	
Add your category	