03-Syntax

August 3, 2022

1 Comments

[18]:
```
# This is a comment
print("something")  # This is an inline comment
```

```
something
```

[19]:
```
# This
# is a
# multiline comment
```

[20]:
```
def myfunction():
    """
    This is an
    example of a
    multiline comment aka docstring

    """



# Three single quotes like this ''' also work
```

[21]:
```
print?
```

```
Docstring:
print(value, …, sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

---

2 Indentation

[22]:
```
# Spacing and indents matters
print("this")
 print("that")
```

```
  Input In [22]
    print("that")
    ^
IndentationError: unexpected indent
```

[23]:
```
a = 33
b = 33

if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

```
a and b are equal
```

[24]:
```
a = 33
b = 33
print("b is greater than a") if b > a else print("a and b are equal") if a == b␣
 ↪else None
```

```
a and b are equal
```

---

3 Escaping

- https://www.w3schools.com/python/gloss_python_escape_characters.asp

[25]:
```
print("Access file here -> c:\newfolder\timmy")
```

```
Access file here -> c:
ewfolder        immy
```

[26]:
```
print(r"Access file here -> c:\newfolder\timmy")
```

```
Access file here -> c:\newfolder\timmy
```

[27]:
```
print("Access file here -> c:\\newfolder\\timmy \nand here -> c:
 ↪\\newfolder\\tommy")
```

```
Access file here -> c:\newfolder\timmy
and here -> c:\newfolder\tommy
```

---

4 Help

[28]:
```
# Display reserved keywords
help("keywords")
```

```
Here is a list of the Python keywords.  Enter any keyword to get more help.

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

[1]:
```
# Zen of Python
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

[ ]:
```

```