Objects and variables

Objects can be stored in variables

```
In [39]: name = "John"
         last_name = "Smith"
         id = "10221"
```

```
In [40]: members = 5
         height = 1.75
```

Objects can also be produced by functions

```
In [4]: name = input("What is your age?")
```

```
In [5]: name
```

```
Out[5]: 'Ardit'
```

```
In [6]: name = input("What is your height?")
```

```
In [9]: name
```

```
Out[9]: '175'
```

Converting to another type

```
In [10]: name = float(input("What is your height?"))
```

```
In [11]: name
```

```
Out[11]: 175.0
```

Functions

Not all functions return a value

```
In [13]: x = print("Hello")
```

```
Hello
```

```
In [14]: x
```

Custom functions can also return or not a value

```
In [15]: def foo():
             value = 10
             return value
```

```
In [16]: x = foo()
```

```
In [17]: x
```

```
Out[17]: 10
```

```
In [20]: def foo():
             value = 10
```

```
In [21]: x = foo()
```

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

Return vs Print

```
In [28]: def foo1():
             value = 10
             return value
```

```
In [29]: def foo2():
             value = 10
             print(value)
```

```
In [30]: x1 = foo1()
```

```
In [31]: x1
```

```
Out[31]: 10
```

```
In [32]: x2 = foo2()
```

```
10
```

```
In [33]: x2
```

Functions with parameters/arguments

```
In [36]: def foo(number):
             result = number * number
             return result
```

```
In [43]: # With argument name
         x = foo(number=10)

In [38]: x

Out[38]: 100

In [41]: # Without argument name
         x = foo(10)
```

Functions with multiple parameters/arguments

```
In [48]: def foo(number1, number2):
             result = number1 * number2
             return result

In [49]: x = foo(10, 20)

In [51]: x

Out[51]: 200
```

Functions with default parameters/arguments

```
In [53]: def foo(number1, number2=2):
             result = number1 * number2
             return result

In [56]: # The default argiment can be ommited
         x = foo(10)

In [57]: x

Out[57]: 20

In [59]: x = foo(10, 3)

Out[59]: 30

In [60]: x

Out[60]: 30
```

Methods

```
In [43]: "hello there".upper()
```

```
Out[43]: 'HELLO THERE'
```

```
In [44]: "hello there".capitalize()
```

```
Out[44]: 'Hello there'
```

```
In [46]: "hello there".title()
```

```
Out[46]: 'Hello There'
```

```
In [63]: greeting = "hello there"
```

```
In [64]: greeting_new = greeting.title()
```

```
Out[64]: 'Hello There'
```

Methods that return an output

```
In [65]: # It returns a new string, but does not modify the original
         word = greeting.title()
         word
```

```
Out[65]: 'Hello There'
```

```
In [69]: # List methods modify the original list
         groceries = ["vinegar", "olives", "bread"]
         varaible = groceries.append("apples")
```

```
In [70]: groceries
```

```
Out[70]: ['vinegar', 'olives', 'bread', 'apples']
```

```
In [71]: groceries.sort()
```

```
In [72]: groceries
```

```
Out[72]: ['apples', 'bread', 'olives', 'vinegar']
```

A list of methods

```
In [74]: dir(str)
```

```
Out[74]: ['__add__',
          '__class__',
          '__contains__',
          '__delattr__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getnewargs__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__init__',
          '__init_subclass__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mod__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__rmod__',
          '__rmul__',
          '__setattr__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'capitalize',
          'casefold',
          'center',
          'count',
          'encode',
          'endswith',
          'expandtabs',
          'find',
          'format',
          'format_map',
          'index',
          'isalnum',
          'isalpha',
          'isascii',
          'isdecimal',
          'isdigit',
          'isidentifier',
          'islower',
          'isnumeric',
          'isprintable',
          'isspace',
          'istitle',
          'isupper',
          'join',
          'ljust',
          'lower',
          'lstrip',
          'maketrans',
          'partition',
          'removeprefix',
          'removesuffix',
          'replace',
          'rfind',
          'rindex',
          'rjust',
          'rpartition',
          'rsplit',
          'rstrip',
          'split',
          'splitlines',
          'startswith',
          'strip',
          'swapcase',
          'title',
          'translate',
          'upper',
          'zfill']
```

In [75]:
```
dir("hello")
```

Out[75]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

```
In [76]: dir(list)
```

```
Out[76]: ['__add__',
          '__class__',
          '__class_getitem__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__getstate__',
          '__gt__',
          '__hash__',
          '__iadd__',
          '__imul__',
          '__init__',
          '__init_subclass__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__reversed__',
          '__rmul__',
          '__setattr__',
          '__setitem__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'append',
          'clear',
          'copy',
          'count',
          'extend',
          'index',
          'insert',
          'pop',
          'remove',
          'reverse',
          'sort']
```

How to create new methods?

First, you need to learn how to create classes.

Lists and tuples

```
In [104…  groceries = ["vinegar", "olives", "bread"]
```

```
In [105…  values = (1920, 1080, "grayscale", "JPG")
```

```
In [ ]:   # Like strings, tuples also have no methods that modify the original
          values.append()
```

Indexing

```
In [82]:  string = "vinegar"
          groceries = ["vinegar", "olives", "bread"]
          values = (1920, 1080, "grayscale", "JPG")
```

```
In [ ]:
```

```
In [27]:  groceries[0]

Out[27]:  'vinegar'
```

```
In [28]:  groceries[2]

Out[28]:  'bread'
```

```
In [29]:  values[2]

Out[29]:  'grayscale'
```

```
In [31]:  string[2]

Out[31]:  'n'
```

```
In [35]:  string[-2]

Out[35]:  'a'
```

```
In [33]:  values[1:3]

Out[33]:  (1080, 'grayscale')
```

```
In [37]:  values[-3:-1]

Out[37]:  (1080, 'grayscale')
```

Dictionaries

```
In [84]: john = {"first name": "John", "last name": "smith", "age":40}
```

```
In [88]: persons1 = [{"first name": "John", "last name": "smith", "age":40},
                     {"first name": "laura", "last name": "eager", "age":45},
                     {"first name": "sim", "last name": "agraval", "age":42}]
```

```
In [89]: persons2 = {"first name": ["john", "laura", "sim"],
                     "last name": ["smith", "eager", "age"],
                     "age": [40, 45, 42]}
```

```
In [87]: john["last name"]
```

```
Out[87]: 'smith'
```

```
In [90]: persons1[2]["first name"]
```

```
Out[90]: 'sim'
```

```
In [91]: persons["first name"][2]
```

```
Out[91]: 'sim'
```

Code blocks

While-Loops

```
In [ ]: while True:
            password = input("Enter password: ")
```

```
In [99]: while password != "pass1":
             password = input("Enter password: ")

         print("Password is correct")
```

```
Password is correct
```

For-Loops

```
In [100…: usernames = ["john", "sim", "spongy"]
          for username in usernames:
              print(username.capitalize())
```

```
John
Sim
Spongy
```

Match-Case

In [101…

```
username = input("Enter username: ")

match username:
    case "john":
        print("Welcome Admin")
    case "sim":
        print("Welcome User")
    case "spongy":
        print("Welcome Guest")
    case _:
        print("Invalid username")
```

```
Welcome User
```

If-Elif-Else

In [14]:

```
password = "pass"
if len(password) > 3:
    print("Password is strong")
else:
    print("Password is weak")
```

```
Password is strong
```

In [12]:

```
password = "pass"

if len(password) > 3:
    print("Password is strong")
elif len(username)==4:
    print("Password is medium")
else:
    print("Password is weak")
```

```
Password is strong
```

f-strings

In [2]:

```
first_name = "naya"
last_name = "anand"
message = f"Hello {first_name.capitalize()} {last_name.capitalize()}! Hav
```

In [3]:

```
message
```

Out[3]:

```
'Hello Naya Anand! Have a nice day!'
```

External Files

Creating files

In [5]:
```
with open("book.txt", "w") as file:
    file.write("Hello there!")
```

In [6]:
```
content = """Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Sed viverra varius lorem sed convallis. Ut finibus arcu ac sem porta soda
Nullam ut eleifend lacus. Sed et aliquam metus.
"""

with open("book.txt", "w") as file:
    file.write(content)
```

In [8]:
```
with open("weather.txt", "w") as file:
    file.writelines(["Clouds\n", "Sun\n", "Sun\n", "Rain\n"])
```

Reading files

In [9]:
```
with open("book.txt", "r") as file:
    content = file.read()
```

In [10]:
```
content
```

Out[10]:
```
'\nLorem ipsum dolor sit amet, consectetur adipiscing elit. \nSed viverr
a varius lorem sed convallis. \nUt finibus arcu ac sem porta sodales. Nu
llam ut eleifend lacus. \nSed et aliquam metus.\n'
```

In [11]:
```
print(content)
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Sed viverra varius lorem sed convallis.
Ut finibus arcu ac sem porta sodales. Nullam ut eleifend lacus.
Sed et aliquam metus.
```

In [15]:
```
with open("weather.txt", "r") as file:
    content = file.readlines()
```

In [16]:
```
content
```

Out[16]:
```
['Clouds\n', 'Sun\n', 'Sun\n', 'Rain\n']
```

List Comprehensions

```
In [18]: clean_content = [item.strip("\n") for item in content]
```

```
In [19]: clean_content
```

```
Out[19]: ['Clouds', 'Sun', 'Sun', 'Rain']
```

Errors

Syntax Errors

```
In [27]: clean_content = [item.strip("\n") for item in content)
```

```
  Cell In [27], line 1
    clean_content = [item.strip("\n") for item in content)
                                                         ^
SyntaxError: closing parenthesis ')' does not match opening parenthesis
'['
```

The error message is not always clear

```
In [28]: clean_content = [item,strip("\n") for item in content]
```

```
  Cell In [28], line 1
    clean_content = [item,strip("\n") for item in content]
                     ^
SyntaxError: did you forget parentheses around the comprehension target?
```

Exceptions

```
In [29]: clean_content = [item.strip("\n") for item in apple]
```

```
------------------------------------------------------------------------
---
NameError                                 Traceback (most recent call la
st)
Cell In [29], line 1
----> 1 clean_content = [item.strip("\n") for item in apple]

NameError: name 'apple' is not defined
```

```
In [31]: clean_content = [item.streap("\n") for item in content]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In [31], line 1
----> 1 clean_content = [item.streap("\n") for item in content]

Cell In [31], line 1, in <listcomp>(.0)
----> 1 clean_content = [item.streap("\n") for item in content]

AttributeError: 'str' object has no attribute 'streap'
```

In [36]:
```
year_of_birth = int(input("Enter the year: "))
current_year = 3221
age = current_year - year_of_birth
print(age)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In [36], line 1
----> 1 year_of_birth = int(input("Enter the year: "))
      2 current_year = 3221
      3 age = current_year - year_of_birth

ValueError: invalid literal for int() with base 10: '12.12.3001'
```

In [44]:
```
current_year = 3221 # Put code like this outside if you can

try:
    year_of_birth = int(input("Enter the year: "))
    age = current_year - year_of_birth
    print(age)
except ValueError:
    print("The format should be YYYY")
```

```
The format should be YYYY
```

Try-except does not catch syntax errors

In [47]:
```
current_year = 3221 # Put code like this outside if you can

try:
    year_of_birth = int(input("Enter the year: ")
    age = current_year - year_of_birth
    print(age # Missing parenthesis
except:
    print("The format should be YYYY")
```

```
  Cell In [47], line 4
    year_of_birth = int(input("Enter the year: ")
                       ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

When to use try-except and when to use if-elif-else

In [46]:
```
current_year = 3221 # Put code like this outside if you can

try:
    year_of_birth = int(input("Enter the year: "))
    age = current_year - year_of_birth
    if age < 150:
        print(age)
    else:
        print("Age too big")
except:
    print("The format should be YYYY")
```

```
Age too big
```

Comments and doc strings

In [48]:
```
def area(a, b):
    """Calculate the area of a rectangle
    given its two sides
    """
    return a * b

rectangle_area = area(10, 20)
```

Modules

In [ ]:
```
import myfile

rectangle_area = myfile.area(10, 20)
```

Standard libraries

In [50]:
```
import glob
import requests
```

In [51]:
```
glob.glob("*.txt")
```

Out[51]:
```
['weather.txt', 'book.txt']
```

```
In [52]: response = requests.get("https://example.com")
         content = response.text
```

```
In [53]: content
```

```
Out[53]: '<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n
<meta charset="utf-8" />\n    <meta http-equiv="Content-type" content="t
ext/html; charset=utf-8" />\n    <meta name="viewport" content="width=de
vice-width, initial-scale=1" />\n    <style type="text/css">\n    body {
\n        background-color: #f0f0f2;\n        margin: 0;\n        paddin
g: 0;\n        font-family: -apple-system, system-ui, BlinkMacSystemFont
, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-seri
f;\n        \n    }\n    div {\n        width: 600px;\n        margin: 5
em auto;\n        padding: 2em;\n        background-color: #fdfdff;\n
border-radius: 0.5em;\n        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.
02);\n    }\n    a:link, a:visited {\n        color: #38488f;\n        t
ext-decoration: none;\n    }\n    @media (max-width: 700px) {\n        d
iv {\n            margin: 0 auto;\n            width: auto;\n        }\n
}\n    </style>    \n</head>\n\n<body>\n<div>\n    <h1>Example Domain</h
1>\n    <p>This domain is for use in illustrative examples in documents.
You may use this\n    domain in literature without prior coordination or
asking for permission.</p>\n    <p><a href="https://www.iana.org/domains
/example">More information...</a></p>\n</div>\n</body>\n</html>\n'
```

Third party libraries

```
In [ ]: pip insall library_name
```

Web apps

```
In [ ]: streamlit
```

Desktop GUI app

```
In [ ]: PySimpleGUI
```