

Wykład8

December 11, 2019

1 Praca z Plikami

```
[ ]: help(open)
```

```
[3]: import os  
os.path.isfile("plik1.txt") #sprawdzamy czy istnieje plik "plik1.txt"
```

```
[3]: False
```

```
[4]: f1 = open("plik1.txt","w")
```

```
[5]: os.path.isfile("plik1.txt")
```

```
[5]: True
```

```
[6]: f1.name
```

```
[6]: 'plik1.txt'
```

```
[7]: f1.mode
```

```
[7]: 'w'
```

```
[8]: f1.write("Początek pliku")
```

```
[8]: 14
```

```
[9]: f1.flush()
```

```
[10]: f1.write("\nDruga linia")
```

```
[10]: 12
```

```
[11]: f1.flush()
```

```
[12]: f1.write("\n0123456789abcdef")
```

[12]: 17

```
[13]: f1.close()
```

```
[14]: f1 = open("plik1.txt", "r+")
      print(f1.read())
      f1.close()
```

Początek pliku
Druga linia
0123456789abcdef

```
[15]: f1 = open("plik1.txt", "r+")
      print(f1.read(14))
```

Początek pliku

```
[16]: print(f1.read(12))
```

Druga linia

```
[17]: f1.tell()
```

[17]: 27

```
[18]: f1.seek(15)
      print(f1.read(12))
```

Druga linia

```
[19]: os.SEEK_SET
```

[19]: 0

```
[ ]: f1.seek(f1.tell() - 12, os.SEEK_SET) #odpowiednik f1.seek(-12,1)
      print(f1.read(12))
```

```
[20]: f1.seek(0, os.SEEK_END) #odpowiednik f1.seek(0,2)
      f1.write("Python")
```

[20]: 6

```
[21]: f1.tell()
```

[21]: 50

```
[22]: f1.close()
```

```
[23]: f1 = open("plik1.txt", "r+b")  
print(f1.read(15))
```

b'Pocz\xc4\x85tek pliku'

```
[24]: print(f1.read(12).decode('utf-8'))
```

Druga linia

```
[25]: f1.seek(-12,1)  
print(f1.read(12).decode('utf-8'))
```

Druga linia

```
[26]: f1.seek(-6,2)  
print(f1.read(6).decode('utf-8'))
```

Python

```
[ ]: f1.close()
```

```
[27]: f1 = open("plik1.txt", "r+")  
print(f1.readlines())  
f1.close()
```

['Początek pliku\n', 'Druga linia\n', '0123456789abcdefPython']

```
[28]: f1 = open("plik1.txt", "r+")  
print(f1.readline())  
print(f1.readline())  
print(f1.readline())  
f1.close()
```

Początek pliku

Druga linia

0123456789abcdefPython

```
[29]: f1 = open("plik1.txt", "a") #lub f1 = open("plik1.txt", "r+") f1.seek(0, os.  
    ↪ SEEK_END)  
f1.writelines(["\nPierwsza nowa linia","\nDruga nowa linia","\nTrzecia nowa_  
    ↪ linia"])  
f1.close()
```

```
[30]: f1 = open("plik1.txt", "r+")
      print(f1.read())
      f1.close()
```

Początek pliku
Druga linia
0123456789abcdefPython
Pierwsza nowa linia
Druga nowa linia
Trzecia nowa linia

```
[31]: f1 = open("plik1.txt", "r+")
      f1.truncate(50)
      f1.seek(0)
      print(f1.read())
      f1.close()
```

Początek pliku
Druga linia
0123456789abcdefPython

1.1 Moduł pickle

```
[32]: import pickle
```

```
[33]: data = {
      'a': [1, 2.0, 3, 4+6j],
      'b': ("character string", b"byte string"),
      'c': {None, True, False}
      }
```

```
[34]: f=open('data_pickle2.txt', 'wb')
      pickle.dump(data, f)
      f.close()
```

```
[35]: data2 = pickle.load(open('data_pickle2.txt', 'rb'))
```

```
[36]: data2
```

```
[36]: {'a': [1, 2.0, 3, (4+6j)],
      'b': ('character string', b'byte string'),
      'c': {False, None, True}}
```

```
[37]: data2["a"]
```

```
[37]: [1, 2.0, 3, (4+6j)]
```

2 Błędy i wyjątki

```
[38]: def iloraz(a, b):  
      """Zwraca a / b lub zero, jeśli b = 0."""  
      if b == 0: #nasz pierwszy "wyjątek"  
          return 0  
      return a / b
```

```
[40]: iloraz(4,2)
```

```
[40]: 2.0
```

Ale można lepiej

```
[41]: def iloraz2(a, b):  
      """Zwraca a / b lub zero, jeśli b = 0."""  
      try: # spróbuj  
          return a / b  
      except: # jeśli error to  
          return 0
```

```
[44]: iloraz2(4, 'a')
```

```
[44]: 0
```

```
[45]: while 1:  
      try:  
          a=int(input("Podaj liczbę:"))  
          print(a)  
          break  
      except ValueError:  
          print("Podana liczba nie jet prawidłowa")
```

```
Podaj liczbę:a  
Podana liczba nie jet prawidłowa  
Podaj liczbę:q  
Podana liczba nie jet prawidłowa  
Podaj liczbę:2  
2
```

```
[46]: int("a")
```

```
↳ -----  
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-46-d9136db7b558> in <module>
----> 1 int("a")
```

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

```
[47]: 10/0
```

```
-----
ZeroDivisionError                                Traceback (most recent call last)
```

```
<ipython-input-47-e574edb36883> in <module>
----> 1 10/0
```

```
ZeroDivisionError: division by zero
```

```
[48]: while 1:
      try:
          a=int(input("Podaj liczbę a:"))
          b=int(input("Podaj liczbę b:"))
          print("Twój wynik to ",a / b)
          break
      except ValueError:
          print("Podana liczba nie jest prawidłowa")
      except ZeroDivisionError:
          print("Nie można dzielić przez 0")
      except: #dowolny inny Error
          print("Coś poszło nie tak...")
```

```
Podaj liczbę a:q
Podana liczba nie jest prawidłowa
Podaj liczbę a:4
Podaj liczbę b:0
Nie można dzielić przez 0
Coś poszło nie tak...
Podaj liczbę a:6
Podaj liczbę b:3
Twój wynik to  2.0
```

Lista wyjątków: <https://docs.python.org/3/library/exceptions.html#builtin-exceptions>

```
[49]: a, b, n = "Python", 2, 1
while n<2:
    try:
        a=int(input("Podaj liczbę a:"))
        b=int(input("Podaj liczbę b:"))
        print("iloraz =",a / b)
        n+=1
    except ValueError:
        print("Podana liczba nie jet prawidłowa")
    except ZeroDivisionError:
        print("Nie mozna dzielic przez 0")
    except: #dowolny inny Error
        print("Coś poszło nie tak...")
print("iloczyn = ",a*b)
```

```
Podaj liczbę a:q
Podana liczba nie jet prawidłowa
iloczyn = PythonPython
Podaj liczbę a:6
Podaj liczbę b:0
Nie mozna dzielic przez 0
iloczyn = 0
Podaj liczbę a:6
Podaj liczbę b:3
iloraz = 2.0
iloczyn = 18
```

```
[50]: a, b, n = "Python", 2, 1
while n<2:
    try:
        a=int(input("Podaj liczbę a:"))
        b=int(input("Podaj liczbę b:"))
        print("iloraz =",a / b)
        n+=1
    except ValueError:
        print("Podana liczba nie jet prawidłowa")
    except ZeroDivisionError:
        print("Nie mozna dzielic przez 0")
    except: #dowolny inny Error
        print("Coś poszło nie tak...")
    else:
        print("iloczyn = ",a*b)
```

```
Podaj liczbę a:q
Podana liczba nie jet prawidłowa
Podaj liczbę a:4
Podaj liczbę b:0
Nie mozna dzielic przez 0
```

```
Podaj liczbę a:4
Podaj liczbę b:2
iloraz = 2.0
iloczyn = 8
```

```
[51]: a, b, n = "Python", 2, 1
while n:
    try:
        a=int(input("Podaj liczbę a:"))
        b=int(input("Podaj liczbę b:"))
        print("iloraz =",a / b)
    except (ValueError, ZeroDivisionError) as naszError:
        print("Błąd: ",naszError)
    except: #dowolny inny Error
        print("Coś poszło nie tak...")
    else:
        print("iloczyn = ",a*b)
    finally:
        n=int(input("Wciśnij '0' aby zakonczyc: "))
```

```
Podaj liczbę a:q
Błąd: invalid literal for int() with base 10: 'q'
Wciśnij '0' aby zakonczyc: 2
Podaj liczbę a:6
Podaj liczbę b:0
Błąd: division by zero
Wciśnij '0' aby zakonczyc: 2
Podaj liczbę a:6
Podaj liczbę b:3
iloraz = 2.0
iloczyn = 18
Wciśnij '0' aby zakonczyc: 0
```

2.1 Zgłaszanie wyjątków

```
[52]: def iloraz(a, b):
      """Zwraca a / b."""
      if b == 0:
          raise NameError("Dzielenie przez zero.")
      return a / b
```

```
[53]: iloraz(6,2)
```

```
[53]: 3.0
```

```
[54]: iloraz(6,0)
```



```

      □
      ↪ -----

NameError                                Traceback (most recent call last)

<ipython-input-54-2c5bf2afd672> in <module>
----> 1 iloraz(6,0)

<ipython-input-52-4004d834b471> in iloraz(a, b)
      2     """Zwraca a / b."""
      3     if b == 0:
----> 4         raise NameError("Dzielenie przez zero.")
      5     return a / b

NameError: Dzielenie przez zero.

```

```

[55]: try:
      iloraz(10, 0)
      except NameError as err:
          print("Błąd:", err)

```

Błąd: Dzielenie przez zero.

2.2 Zastosowanie do pracy z plikami

```

[ ]: try:
      file = open("Wadliwy_plik")
      data = file.read() # zgłasza wyjątek
      finally:           # porządek nawet w przypadku wyjątku
          print("Pracuje dalej...")
          file.close()

```

Ale lepiej tak

```

[ ]: with open("plik_z_danymi") as file:
      data = file.read()

```

3 Debagowanie

```
[ ]: # program fibanccci_deb.py
fibo=[0,1]
x,y=0,1
i=0
while y<20:
    #print(y)
    if not __debug__:
        print("i=",i)
        print("fibo =",fibo)
        print("fibo[i-2] =", x)
        print("fibo[i-1] =", y)
        print()
        i+=1
    x,y = y,x+y
    fibo.append(y)

if __debug__:
    print(fibo)
```

Sprawdźmy: python -O fibanccci_deb.py

Polecam też sprawdzić modul pdb !!!


3.1 Assert

```
[ ]: assert <test>, <działanie> # Część <działanie> jest opcjonalna
                                     #działa tak samo jak poniższy kod:

if __debug__:
    if not <test>:
        raise AssertionError(<dane>)
```

```
[56]: def f(x):
        assert x < 0, 'x musi być ujemne'
        return x ** 2
```

```
[57]: f(2)
```

 -----

AssertionError

Traceback (most recent call last)

```
<ipython-input-57-c510dc86724b> in <module>
----> 1 f(2)
```

```
<ipython-input-56-8cd8d6e97a66> in f(x)
     1 def f(x):
----> 2     assert x < 0, 'x musi być ujemne'
     3     return x ** 2
```

```
AssertionError: x musi być ujemne
```

```
[58]: f(-2)
```

```
[58]: 4
```

Przetestuj assert_test.py