PY: Podstawy Ściągawka

main.py

```
if ( 5 > 2 ):
    print("greater than")

if ( 5 > 2 ):
    print("greater than")


x = 56
y = 56.78
w = 'a'
z = "string"

print("x: "+str(x))
print("y: "+str(y))
#print("y: "+y)
#TypeError: can only concatenate str (not "float") to str

print(w + z)

print("data", end="")

print(x); print(y); print(z)

print(4+2)

print("XYZ", x, y, z)

"""
Komentarz wieloliniowy
"""

print(type(x))
print(type(str(x)))

print(type(str(y)))


print(type(w))
print(type(z))


camelCase = 0
snake_case = 0
kebabcase = 0
PascalCase = 0
```

```
a, b, c = "a", "b", "c"

a = b = c = "xyz"

a = [1,2,3,4,5]
a.append(6)
print(a)
print(a[1])



def myfunc():
    global glbl
    glbl  = "this is global variable"
    print("Python is ", x)

myfunc()
print(glbl)


"""
Text Type:  str
Numeric Types:  int, float, complex
Sequence Types:     list, tuple, range
Mapping Type:   dict
Set Types:  set, frozenset
Boolean Type:   bool
Binary Types:   bytes, bytearray, memoryview
None Type:  NoneType
"""

def multiply(a,b):
    c = a * b
    return(c)

print(multiply(10,2))

b = "Hello, World!"
print(b[2:5])

b = "Hello, World!"
print(b[:5])

b = "Hello, World!"
print(b[2:])

b = "Hello, World!"
print(b[-5:-2])

a = "Hello, World!"
```

```
print(a.upper())

a = "Hello, World!"
print(a.lower())

a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"

a = "Hello, World!"
print(a.replace("H", "J"))

a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']

a = "Hello"
b = "World"
c = a + b
print(c)

age = 36
txt = f"My name is John, I am {age}"
print(txt)

price = 59
txt = f"The price is {price} dollars"
print(txt)

price = 59
txt = f"The price is {price:.2f} dollars"
print(txt)

txt = f"The price is {20 * 59} dollars"
print(txt)

txt = "We are the so-called \"Vikings\" from the north."
print(txt)


#String Methods
"""
Method  Description
capitalize()    Converts the first character to upper case
casefold()  Converts string into lower case
center()    Returns a centered string
count() Returns the number of times a specified value occurs in a
string
encode()    Returns an encoded version of the string
endswith()  Returns true if the string ends with the specified value
expandtabs()    Sets the tab size of the string
find()  Searches the string for a specified value and returns the
position of where it was found
```

```
format()    Formats specified values in a string
format_map()    Formats specified values in a string
index() Searches the string for a specified value and returns the
position of where it was found
isalnum()   Returns True if all characters in the string are
alphanumeric
isalpha()   Returns True if all characters in the string are in the
alphabet
isascii()   Returns True if all characters in the string are ascii
characters
isdecimal() Returns True if all characters in the string are decimals
isdigit()   Returns True if all characters in the string are digits
isidentifier()  Returns True if the string is an identifier
islower()   Returns True if all characters in the string are lower case
isnumeric() Returns True if all characters in the string are numeric
isprintable()   Returns True if all characters in the string are
printable
isspace()   Returns True if all characters in the string are
whitespaces
istitle()   Returns True if the string follows the rules of a title
isupper()   Returns True if all characters in the string are upper case
join()  Joins the elements of an iterable to the end of the string
ljust() Returns a left justified version of the string
lower() Converts a string into lower case
lstrip()    Returns a left trim version of the string
maketrans() Returns a translation table to be used in translations
partition() Returns a tuple where the string is parted into three parts
replace()   Returns a string where a specified value is replaced with a
specified value
rfind() Searches the string for a specified value and returns the last
position of where it was found
rindex()    Searches the string for a specified value and returns the
last position of where it was found
rjust() Returns a right justified version of the string
rpartition()    Returns a tuple where the string is parted into three
parts
rsplit()    Splits the string at the specified separator, and returns a
list
rstrip()    Returns a right trim version of the string
split() Splits the string at the specified separator, and returns a
list
splitlines()    Splits the string at line breaks and returns a list
startswith()    Returns true if the string starts with the specified
value
strip() Returns a trimmed version of the string
swapcase()  Swaps cases, lower case becomes upper case and vice versa
title() Converts the first character of each word to upper case
translate() Returns a translated string
upper() Converts a string into upper case
zfill() Fills the string with a specified number of 0 values at the
beginning
```

```
"""
print(10 > 9)
print(10 == 9)
print(10 < 9)


if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")

print(bool("Hello"))
print(bool(15))

def myFunction() :
  return True

print(myFunction())

if myFunction():
  print("YES!")
else:
  print("NO!")

x = 200
print(isinstance(x, int))

print(12 % 5)
print(2 ** 3)
print(10 // 3)

fruits = ["apple", "banana", "cherry"]

print("banana" in fruits)

x = 5

print(x > 0 and x < 10)

x = 5

print(x < 5 or x > 10)

x = 5

print(not(x > 3 and x < 10))

mytuple = ("apple", "banana", "cherry")

print(mytuple)
```

```
tuple1 = ("abc", 34, True, 40, "male")

print(tuple1)

list = ["abx",123]

print(list)

thistuple = tuple(("apple", "banana", "cherry")) # note the double
round-brackets
print(thistuple)

thisset = {"apple", "banana", "cherry"}
print(thisset)

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "colors": ["red", "white", "blue"]
}

print(thisdict)
print(thisdict["brand"])
print(len(thisdict))

thisdict = dict(name = "John", age = 36, country = "Norway")
print(thisdict)

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)

i = 1
while i < 6:
  print(i)
  i += 1

i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1

i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")


fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break

fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)

for x in range(6):
  print(x)

for x in range(2, 6):
  print(x)

for x in range(2, 30, 3):
  print(x)

for x in range(6):
  print(x)
else:
  print("Finally finished!")


adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)

for x in [0, 1, 2]:
  pass

"""
Equals: a == b
Not Equals: a != b
Less than: a < b
Less than or equal to: a <= b
```

```
Greater than: a > b
Greater than or equal to: a >= b
"""

age = 20
if age >= 18:
  print("You are an adult")
  print("You can vote")
  print("You have full legal rights")
```