**Subroutines**

**Before Class**

* Subroutines, and specifically functions, are a convenient way to divide a program code into useful blocks, allowing to make a program code more readable, reuse it and save some time. A function is a block of code which only runs when it is called. There are a number of built-in functions, ready to use, e.g. len(), print(), input() or type().
* <https://docs.python.org/3/library/functions.html>
* Using built-in Python functions, write a program that calculates and displays:
* length of the phrase: "computer science"
* letter read from the keyboard
* string representing the number 5068
* numeric representing the string "20303"
* the smallest number given: 4,7,2,3,9,8
* In addition to the built-in functions, you can use numerous functions available in ready-to-use modules. One example is the 'math' module.
* <https://docs.python.org/3/library/math.html>
* Using functions and constants available in the 'math' module, write a program that calculates and displays:
* natural logarithm of 5
* e raised to the power of 3
* square root of 7
* sine of 90 degrees
* Familiarise yourself with the concept of dividing the program code into a smaller parts that performs specific tasks.
* <https://youtu.be/NE97ylAnrz4?feature=shared>
* From the textbook, read the chapter 4 (Functions).
* Complete all exercises available in the textbook, chapter 4, Exercise section.
* Familiarise yourself with defining anonymous (lambda) functions in Python.
* <https://youtu.be/25ovCm9jKfA?feature=shared>
* Define an anonymous function that calculates the body mass index (BMI) for the given weight in kg and height in cm. Then, calculate the BMI for Peter (81kg, 182cm). Sample result:
* Peter’s BMI is …
* If you use a function in a program, variables can appear both inside and outside the function. Variables defined inside a function are usually not visible and cannot be used outside the function. On the other hand, variables defined outside the function are visible and can be used inside the function. Familiarize yourself with the concepts of 'global variable' and 'local variable' and how to use them.
* <https://youtu.be/QYUbLevwgDQ?feature=shared>
* Familiarise yourself with dividing a program code into modules:
* <https://www.w3schools.com/python/python_modules.asp>
* <https://docs.python.org/3/tutorial/modules.html>
* Sometimes we want certain statements contained in a module to execute only when the module is run, and not to execute when the module is imported. Check how this can be achieved.
* <https://youtu.be/NB5LGzmSiCs?feature=shared>
* <https://docs.python.org/3/library/__main__.html>
* Create the following two modules:
* **converters.py**
* def m\_to\_cm(n):  
   return n\*100  
    
  def cm\_to\_m(n):  
   return n/100  
    
  if \_\_name\_\_ == "\_\_main\_\_":  
   # only execute when you run this module  
   print(f'2m = {m\_to\_cm(2)}cm')  
   print(f'532cm = {cm\_to\_m(532)}m')
* **myprogram.py**
* import converters  
  print('## Test converters')  
  print(f'Three meters is {converters.m\_to\_cm(3)}cm')
* First, run converters.py and see what information is displayed. Then, run myprogram.py, paying attention to the information that is displayed. Do you know why the information displayed is different?

**During Class**

**Functions**

* Using built-in Python functions, write a program that calculates and displays:
* the largest number given: 7,5,6,3,8,2
* binary string representing decimal number 304
* hexadecimal string representing decimal number 304
* integer representing the Unicode code of the € sign
* absolute value of -17
* # task a  
  max\_number = max(7,5,6,3,8,2)  
  print('Max number of 7,5,6,3,8,2 is', max\_number)   
    
  # task b  
  # task c  
  # task d  
  # task e
* Using functions and constants available in the 'math' module, write a program that calculates and displays:
* natural logarithm of 5
* e raised to the power of 3
* square root of 7
* sine of 90 degrees
* Define the display\_program\_name() function that displays the name of your study program. Then, write a program that displays the study program four times. Apply a loop statement. Sample result:
* def display\_program\_name():  
   print('Applied informatics')  
    
  for i in range(4):  
   display\_program\_name()
* Define a function phone\_keyboard() that displays numbers in the layout as below (like on a phone keypad). Apply a loop statement. Then, call the function. Sample result:
* 1 2 3  
  4 5 6  
  7 8 9

**Passing and returning values**

* Define a function product(x, y) that displays a product of two numbers. Then, call the function.
* def product(x,y):  
   return x\*y  
    
  a = 3  
  b = 4  
  print(f"The product of {a} and {b} is {product(a,b)}")
* Define the function different(n1,n2,n3), which returns True if all three numbers n1,n2,n3 are different or False otherwise. Then, write a program that reads three integers from the keyboard. Checks whether the numbers are different. Sample result:
* Enter first number: …  
  Enter second number: …  
  Enter third number: …  
  Numbers …, …, and … are different
* Define a function numbers(n) that returns a string containing integer numbers from 1 to n, separated by a single space character. Then, call the function and display numbers from 1 to 15 and from 1 to 7. Sample result:
* Numbers <1,15>: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15  
  Numbers <1,7>: 1 2 3 4 5 6 7

**Modules**

* In a separate module, define a function that calculates the sum of digits. Use the function to calculate the sum of digits entered from the keyboard. To do it, copy the following modules. Then, run the programs digits.py and myprogram.py separately. Try to analyze the results. Do you understand how to import a module and how to call the functions contained in the module?
* **digits.py**
* def sum\_digits(n):  
      sum = 0  
      while n > 0:  
          sum += n % 10  
          n //= 10  
      return sum  
    
  if \_\_name\_\_ == "\_\_main\_\_":  
      # check if function works  
      print(sum\_digits(7182))  
      print(sum\_digits(0))  
      print(sum\_digits(333))
* **myprogram.py**
* import digits  
    
  number = int(input("Enter a number: "))  
  sum\_d = digits.sum\_digits(number)  
  msg = f"Sum of digits {number} is {sum\_d}"  
  print(msg)
* In the module mykeyboard.py, define a function read\_number() that returns an integer number entered from the keyboard. The function should print a text prompting user to enter data 'Enter a number: '. Then, use the function to read two numbers from the keyboard. To test the function, use the \_\_name\_\_ variable. Display the sum of two entered numbers. Sample result:
* Enter a number: 34  
  Enter a number: 7  
  34 + 7 = 41

**After Class**

* In the module mymath.py, define the function generate\_number() that creates and returns random integer number in the range of <1,9>. Then create a main program, in which, first import modules mymath.py and mykeyboard.py, you created earlier. The program is a simple guessing game. The user enters a one-digit number from the keyboard. The computer then generates a random one-digit number. If the numbers match, the user wins the game. Sample result:
* Enter a number: 7  
  Computer number: 7  
  You won the game!!
* Each month of a calendar year can be expressed by its name or by a number that indicates the position of the month in year. In a separate module, define a function month(n) that returns a month name based on the month number (values from 1 to 12). Then, create a program to display the name of the month 7. Import the module with the created function. Sample result:
* Enter month number: 9  
  The name of month 9 is September
* Create a program that calculates how many times the given letter appears in any text. Then create a program and check how many times the letter ‘e’ appears in the text below. In a separate module, define a function for making calculations. Sample result:
* You never get a second chance to make a first impression  
  The number of letter 'e': 7
* In a separate module, define a function that checks if the number is within the range <x, y>. The function returns a boolean value. Then, create a program and use the function you defined. Sample result:
* A number: 7  
  Number 7 in the range <2,15>: yes
* Define an anonymous function that returns True when the first number is greater than the second one. Otherwise returns False. Use a conditional operator. Then, check the function for pairs of numbers: 34, 25 and 19,23.
* Define an anonymous function that returns True when a number is even or False otherwise.
* The credit card number consists of 16 digits. In a separate module, define a function f(card\_number) that masks the card number. The function returns a character string in which only the first two and the last four digits of the card number are visible. The remaining digits of the card number are replaced with an asterisk. Then, create a program that masks some credit card digits. Import the module with the created function. Finally, display the credit card number. Sample result:
* f("5290312400019022") returns "52\*\*\*\*\*\*\*\*\*\*9022"
* The binary numerical system uses two symbols to represent a number: 0 and 1. Define a function f(binary\_number) that returns True if the given string of digits is a valid binary number, or False otherwise. Sample result:
* f("101101") returns True  
  f("1311a10100") returns False
* The vending machine accepts 1, 2 and 5 PLN coins. Define a function f(amount\_to\_pay) that returns the minimum number of coins that can be used to pay for the purchased product. Sample result:
* f(23) returns 6  
  f(8) returns 3  
  f(2) returns 1  
  f(0) returns 0
* Create a function f(number, even) that computes the sum of the digits of a number. When the value of the even parameter is True, the function returns the sum of the even digits. When the value of the even parameter is False, the function returns the sum of the odd digits. Sample result:
* f(3124,True) returns 6  
  f(3124,False) returns 4  
  f(20576,False) returns 12  
  f(20576,True) returns 8  
  f(13115,True) returns 0
* Define the function f(x,y) that returns the number of negative even numbers in the range <x,y>. Sample result:
* f(-7,8) returns 3  
  f(-1,11) returns 0
* Define the function f(n1,n2,n3), which returns True if at least one of the numbers n1,n2,n3 is negative or False otherwise. Sample result:
* f(11,6,-4) returns True  
  f(5,4,14) returns False
* Define a function f(n) that returns a string of n asterisks, separated by a slash sign. Sample result:
* f(4) returns "\*/\*/\*/\*"  
  f(1) returns "\*"
* Define the function f(n), which returns numbers from 1 to n as a string. Sample result:
* f(11) returns "1234567891011"  
  f(4) returns "1234"
* Two numbers and an operator are given. Define a function f(number1,number2,operator) that returns the result of an arithmetic operation. The available operators are +,-,\*,%,\*\*. Sample result:
* f(2,3, "+") returns 5  
  f(2,3, "%") returns 2  
  f(2,3, "\*\*") returns 8  
  f(2,3, "\*") returns 6  
  f(2,3, "-") returns -1
* 33. A device in a door registers people entering and leaving a room. The + sign means a person entering a room and the – sign a person leaving a room. Define the function f(detector) that returns True if at least 3 people were in the room at the same time, or False otherwise. Sample result:
* f("+-+++-+---") returns True  
  f("+-+-+-+-") returns False  
  f("+-++-+--") returns False  
  f("+-++-++-+---") returns True
* Define the function f(n), which returns the n-th value of the Fibonacci sequence. The sequence is defined as follows: the first value of the sequence is 0, the second value is 1. Each subsequent value is the sum of the previous two. Sample result:
* f(5) returns 3  
  f(9) returns 21
* A palindrome is an expression that sounds the same when read backwards. Define a function f(palindrome) that returns True if the expression is a palindrome or False otherwise. Sample result:
* f("radar") returns True  
  f("12-11-21") returns True  
  f("book") returns False
* A sentence is an ordered group of words separated by spaces (spaces). Define a function f(sentence) that returns a sentence with spaces removed. Sample result:
* f("integrated development environment") returns "integrateddevelopmentenvironment"  
  f("A programming language is a system of notation for writing computer programs") returns "Aprogramminglanguageisasystemofnotationforwritingcomputerprograms"
* Define a function f(number) that returns the sum of repeated digits in a number. Sample result:
* f(1027) returns 0  
  f(230335) returns 9  
  f(513553007) returns 21
* Define the function f(n) that returns the n-th prime number. A prime number is a natural number greater than 1, divisible by 1 and that number. Sample result:
* f(1) returns 2  
  f(5) returns 11
* Define the function f(number1,number2,number3), which returns the difference between the largest and smallest numbers. Sample result:
* f(7,4,9) returns 5  
  f(2,12,8) returns 10
* A text contains any number of words. Define a function f(name) that returns the acronym (first letters of all words). Sample result:
* f("Internet of Things") returns "IoT"  
  f("For Your Information") returns "FYI"  
  f("Python") returns "P"
* A valid password should consist of at least six different characters. Define a function f(password) that returns True if the password is correct or False otherwise. Sample result:
* f("ax15") returns False  
  f("book123") returns False  
  f("A2water3") returns True  
  f("qwerty") returns True  
  f("") returns False
* An expression contains operators for adding and subtracting single-digit numbers. Define a function f(expression) that returns the value of the expression. Sample result:
* f("2+3") returns 5  
  f("3+8+1") returns 12  
  f("2+3-4+5-0") returns 6
* Define the function f(x,y), which returns the sum of numbers in the range <x,y> that are completely divisible by 2 and 3 and not divisible by 4. Sample result:
* f(1,20) returns 24  
  f(10,30) returns 48
* Define a function f(text) that returns the given text with all characters separated by "-" (minus sign). Example:
* f("Univesity") returns "U-n-i-v-e-r-s-i-t-y"  
  f("UE") returns "U-E"  
  f("x") returns "x"  
  f("") returns ""
* Products are marked with a special code consisting of 3 digits and a fourth control digit. The forth digit is determined by calculating the remainder of dividing the sum of the first three digits by 7. Define a function f(product\_code) that returns True if the product code is correct or False otherwise. Sample result:
* f("1082") returns True  
  f("2035") returns True  
  f("1114") returns False  
  f("7071") returns False
* The sequence of digits contains the number of points rolled with a dice. Define a function f(dice) that returns a number specifying the number of dice rolled the most times in a row. Sample result:
* f("5233165554211") returns 5  
  f("2133") returns 3
* The following function calculates the factorial recursively. Try to analyse the function. Do you understand how it works? Then, write a program and use the function to calculate the factorial value for n = 5.
* def factorial(n):  
    
   # 0! = 1, 1! = 1  
   if n==0 or n==1:  
   return 1  
    
   # n! = n \* (n-1)!  
   if n > 1:  
   return n \* factorial(n-1)
* Define a function sum(n) that for the given natural number n calculates the sum of all natural numbers between 1 and n. Apply recursion. Then, create a program that calculates the sum of natural numbers in the range <1,10>.
* Define a function power(x, n) that calculates xn. Apply recursion. Then, calculate 53.
* Tip: xn = x \* xn-1