Plotting and Programming with Python at NSBE48

Contents

Notes

Resources

- Shell Lesson
- Python Lesson
- workshop website

This site has the notes for the workshop.

```
4+5 +6
```

```
15
```

Python Basics

The first notebook of the day

Today's lesson covers:

- notebooks
- data analysis
- plotting

hello
goodbye
sarah

next up

Variables and Assignment

```
name = 'sarah'
sentence = "Sarah's title is Dr. "
```

```
name
```

```
'sarah'
```

```
sentence
name
```

```
'sarah'
```

```
last_name
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1857/640889992.py in <module>
----> 1 last_name

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

```
last_name = 'Brown'
```

```
last_name
```

```
'Brown'
```

```
print(name)
```

```
sarah
```

```
name
```

```
'sarah'
```

```
4+5
```

```
9
```

```
age = 34
```

```
age
```

```
34
```

```
age + 5
```

```
39
```

```
full_name = name + last_name
```

```
full_name
```

```
'sarahBrown'
```

```
name*3
```

```
'sarahsarahsarah'
```

```
[1,3,4]*4
```

```
[1, 3, 4, 1, 3, 4, 1, 3, 4, 1, 3, 4]
```

```
name[0]
```

```
's'
```

```
name[5]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-19-94fccd0ebc6d> in <module>
----> 1 name[5]

IndexError: string index out of range
```

```
name[1:3]
```

```
'ar'
```

```
name[0:4]
```

```
'sara'
```

```
name[-1]
```

```
'h'
```

```
my_string = 'NSBE 48 is in Anaheim'
```

How can you print out only the word Anaheim (7characters)

```
my_string[-7:]
```

```
'Anaheim'
```

```
my_string[14:]
```

```
'Anaheim'
```

Notes so far

```
print('before')
print()
print('after')
```

```
before

after
```

Print out what we saw above using only one print call

```
print('before','after',sep='\n\n')
```

```
before

after
```

```
print(In [26])
```

```
print('before')
print()
print('after')
```

```
print?
```

```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    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.
```

```
my_string = 'Hello world!'
```

```
len(my_string)
```

```
12
```

```
my_string.upper
```

```
<function str.upper()>
```

```
my_string.upper()
```

```
'HELLO WORLD!'
```

```
my_string.swapcase()
```

```
'hELLO WORLD!'
```

```
my_string.isupper()
```

```
False
```

```
my_string.upper().isupper()
```

```
True
```

```
remaing = 100-age
```

```
remaing
```

```
66
```

```
name*3
```

```
'sarahsarahsarah'
```

```
4*5
```

```
20
```

```
import math
```

```
math.pi
```

```
3.141592653589793
```

```
math.e
```

```
2.718281828459045
```

```
help(math)
```

```
Help on module math:

NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.8/library/math

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

DESCRIPTION
    This module provides access to the mathematical functions
    defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

        Unlike atan(y/x), the signs of both x and y are considered.

    atanh(x, /)
        Return the inverse hyperbolic tangent of x.

    ceil(x, /)
        Return the ceiling of x as an Integral.

        This is the smallest integer >= x.

    comb(n, k, /)
        Number of ways to choose k items from n items without repetition and
without order.

        Evaluates to n! / (k! * (n - k)!) when k <= n and evaluates
        to zero when k > n.

        Also called the binomial coefficient because it is equivalent
        to the coefficient of k-th term in polynomial expansion of the
        expression (1 + x)**n.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    copysign(x, y, /)
        Return a float with the magnitude (absolute value) of x but the sign of y.

        On platforms that support signed zeros, copysign(1.0, -0.0)
        returns -1.0.

    cos(x, /)
        Return the cosine of x (measured in radians).

    cosh(x, /)
        Return the hyperbolic cosine of x.

    degrees(x, /)
```

```
        Convert angle x from radians to degrees.

    dist(p, q, /)
        Return the Euclidean distance between two points p and q.

        The points should be specified as sequences (or iterables) of
        coordinates.  Both inputs must have the same dimension.

        Roughly equivalent to:
            sqrt(sum((px - qx) ** 2.0 for px, qx in zip(p, q)))

    erf(x, /)
        Error function at x.

    erfc(x, /)
        Complementary error function at x.

    exp(x, /)
        Return e raised to the power of x.

    expm1(x, /)
        Return exp(x)-1.

        This function avoids the loss of precision involved in the direct
evaluation of exp(x)-1 for small x.

    fabs(x, /)
        Return the absolute value of the float x.

    factorial(x, /)
        Find x!.

        Raise a ValueError if x is negative or non-integral.

    floor(x, /)
        Return the floor of x as an Integral.

        This is the largest integer <= x.

    fmod(x, y, /)
        Return fmod(x, y), according to platform C.

        x % y may differ.

    frexp(x, /)
        Return the mantissa and exponent of x, as pair (m, e).

        m is a float and e is an int, such that x = m * 2.**e.
        If x is 0, m and e are both 0.  Else 0.5 <= abs(m) < 1.0.

    fsum(seq, /)
        Return an accurate floating point sum of values in the iterable seq.

        Assumes IEEE-754 floating point arithmetic.

    gamma(x, /)
        Gamma function at x.

    gcd(x, y, /)
        greatest common divisor of x and y

    hypot(...)
        hypot(*coordinates) -> value

        Multidimensional Euclidean distance from the origin to a point.

        Roughly equivalent to:
            sqrt(sum(x**2 for x in coordinates))

        For a two dimensional point (x, y), gives the hypotenuse
        using the Pythagorean theorem:  sqrt(x*x + y*y).

        For example, the hypotenuse of a 3/4/5 right triangle is:

            >>> hypot(3.0, 4.0)
            5.0

    isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
        Determine whether two floating point numbers are close in value.

          rel_tol
            maximum difference for being considered "close", relative to the
            magnitude of the input values
          abs_tol
            maximum difference for being considered "close", regardless of the
            magnitude of the input values
```

```
        Return True if a is close in value to b, and False otherwise.

        For the values to be considered close, the difference between them
        must be smaller than at least one of the tolerances.

        -inf, inf and NaN behave similarly to the IEEE 754 Standard.  That
        is, NaN is not close to anything, even itself.  inf and -inf are
        only close to themselves.

    isfinite(x, /)
        Return True if x is neither an infinity nor a NaN, and False otherwise.

    isinf(x, /)
        Return True if x is a positive or negative infinity, and False otherwise.

    isnan(x, /)
        Return True if x is a NaN (not a number), and False otherwise.

    isqrt(n, /)
        Return the integer part of the square root of the input.

    ldexp(x, i, /)
        Return x * (2**i).

        This is essentially the inverse of frexp().

    lgamma(x, /)
        Natural logarithm of absolute value of Gamma function at x.

    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.

        If the base not specified, returns the natural logarithm (base e) of x.

    log10(x, /)
        Return the base 10 logarithm of x.

    log1p(x, /)
        Return the natural logarithm of 1+x (base e).

        The result is computed in a way which is accurate for x near zero.

    log2(x, /)
        Return the base 2 logarithm of x.

    modf(x, /)
        Return the fractional and integer parts of x.

        Both results carry the sign of x and are floats.

    perm(n, k=None, /)
        Number of ways to choose k items from n items without repetition and with
order.

        Evaluates to n! / (n - k)! when k <= n and evaluates
        to zero when k > n.

        If k is not specified or is None, then k defaults to n
        and the function returns n!.

        Raises TypeError if either of the arguments are not integers.
        Raises ValueError if either of the arguments are negative.

    pow(x, y, /)
        Return x**y (x to the power of y).

    prod(iterable, /, *, start=1)
        Calculate the product of all the elements in the input iterable.

        The default start value for the product is 1.

        When the iterable is empty, return the start value.  This function is
        intended specifically for use with numeric values and may reject
        non-numeric types.

    radians(x, /)
        Convert angle x from degrees to radians.

    remainder(x, y, /)
        Difference between x and the closest integer multiple of y.

        Return x - n*y where n*y is the closest integer multiple of y.
        In the case where x is exactly halfway between two multiples of
        y, the nearest even value of n is used. The result is always exact.

    sin(x, /)
```

```
        Return the sine of x (measured in radians).

    sinh(x, /)
        Return the hyperbolic sine of x.

    sqrt(x, /)
        Return the square root of x.

    tan(x, /)
        Return the tangent of x (measured in radians).

    tanh(x, /)
        Return the hyperbolic tangent of x.

    trunc(x, /)
        Truncates the Real x to the nearest Integral toward 0.

        Uses the __trunc__ magic method.

DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
    tau = 6.283185307179586

FILE
    /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/lib-
dynload/math.cpython-38-darwin.so
```

```
help(math.pi)
```

```
Help on float object:

class float(object)
 |  float(x=0, /)
 |
 |  Convert a string or number to a floating point number, if possible.
 |
 |  Methods defined here:
 |
 |  __abs__(self, /)
 |      abs(self)
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __divmod__(self, value, /)
 |      Return divmod(self, value).
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __float__(self, /)
 |      float(self)
 |
 |  __floordiv__(self, value, /)
 |      Return self//value.
 |
 |  __format__(self, format_spec, /)
 |      Formats the float according to format_spec.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getnewargs__(self, /)
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __hash__(self, /)
 |      Return hash(self).
 |
 |  __int__(self, /)
 |      int(self)
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
```

```
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __mod__(self, value, /)
 |      Return self%value.
 |
 |  __mul__(self, value, /)
 |      Return self*value.
 |
 |  __ne__(self, value, /)
 |      Return self!=value.
 |
 |  __neg__(self, /)
 |      -self
 |
 |  __pos__(self, /)
 |      +self
 |
 |  __pow__(self, value, mod=None, /)
 |      Return pow(self, value, mod).
 |
 |  __radd__(self, value, /)
 |      Return value+self.
 |
 |  __rdivmod__(self, value, /)
 |      Return divmod(value, self).
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  __rfloordiv__(self, value, /)
 |      Return value//self.
 |
 |  __rmod__(self, value, /)
 |      Return value%self.
 |
 |  __rmul__(self, value, /)
 |      Return value*self.
 |
 |  __round__(self, ndigits=None, /)
 |      Return the Integral closest to x, rounding half toward even.
 |
 |      When an argument is passed, work like built-in round(x, ndigits).
 |
 |  __rpow__(self, value, mod=None, /)
 |      Return pow(value, self, mod).
 |
 |  __rsub__(self, value, /)
 |      Return value-self.
 |
 |  __rtruediv__(self, value, /)
 |      Return value/self.
 |
 |  __sub__(self, value, /)
 |      Return self-value.
 |
 |  __truediv__(self, value, /)
 |      Return self/value.
 |
 |  __trunc__(self, /)
 |      Return the Integral closest to x between 0 and x.
 |
 |  as_integer_ratio(self, /)
 |      Return integer ratio.
 |
 |      Return a pair of integers, whose ratio is exactly equal to the original
float
 |      and with a positive denominator.
 |
 |      Raise OverflowError on infinities and a ValueError on NaNs.
 |
 |      >>> (10.0).as_integer_ratio()
 |      (10, 1)
 |      >>> (0.0).as_integer_ratio()
 |      (0, 1)
 |      >>> (-.25).as_integer_ratio()
 |      (-1, 4)
 |
 |  conjugate(self, /)
 |      Return self, the complex conjugate of any float.
 |
 |  hex(self, /)
 |      Return a hexadecimal representation of a floating-point number.
 |
 |      >>> (-0.1).hex()
 |      '-0x1.999999999999ap-4'
 |      >>> 3.14159.hex()
```

```
 |      '0x1.921f9f01b866ep+1'
 |
 |  is_integer(self, /)
 |      Return True if the float is an integer.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __getformat__(typestr, /) from builtins.type
 |      You probably don't want to use this function.
 |
 |        typestr
 |          Must be 'double' or 'float'.
 |
 |      It exists mainly to be used in Python's test suite.
 |
 |      This function returns whichever of 'unknown', 'IEEE, big-endian' or 'IEEE,
 |      little-endian' best describes the format of floating point numbers used by
the
 |      C type named by typestr.
 |
 |  __set_format__(typestr, fmt, /) from builtins.type
 |      You probably don't want to use this function.
 |
 |        typestr
 |          Must be 'double' or 'float'.
 |        fmt
 |          Must be one of 'unknown', 'IEEE, big-endian' or 'IEEE, little-endian',
 |          and in addition can only be one of the latter two if it appears to
 |          match the underlying C reality.
 |
 |      It exists mainly to be used in Python's test suite.
 |
 |      Override the automatic determination of C-level floating point type.
 |      This affects how floats are converted to and from binary strings.
 |
 |  fromhex(string, /) from builtins.type
 |      Create a floating-point number from a hexadecimal string.
 |
 |      >>> float.fromhex('0x1.ffffp10')
 |      2047.984375
 |      >>> float.fromhex('-0x1p-1074')
 |      -5e-324
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  imag
 |      the imaginary part of a complex number
 |
 |  real
 |      the real part of a complex number
```

```
math.cos(2*math.pi)
```

```
1.0
```

```
from math import cos, pi
```

```
cos(pi)
```

```
-1.0
```

```
import math as m
```

```
m.sin(m.pi)
```

```
1.2246467991473532e-16
```

Put these in order:

```
bases="ACTTGCTTGAC"
import math
import random
___ = random.randrange(n_bases)
___ = len(bases)
print("random base ", bases[___], "base index", ___)
```

```
# this is a comment

# import the libraries
import math
import random

# list of bases
bases="ACTTGCTTGAC"

#  number of bases
n_bases= len(bases)

# sample a random integer from 0 to the number of bases
rand_index = random.randrange(n_bases)

# print by selecting the random index
#     and print out that number too
print("random base ", bases[rand_index], "base index", rand_index)
```

```
random base  G base index 8
```

```
bases[8]
```

```
'G'
```

By Sarah M Brown, Jason Williams