NumPy Tutorial Cheat Sheet

I. NumPy Tutorial (Core Basics)

np.array()
Creates an array from a list or tuple.

```
arr = np.array([1, 2, 3])
print(arr)
# Output: [1 2 3]
```

Indexing []
Accesses a specific element by its position.

```
arr = np.array([10, 20, 30])
print(arr[1])
# Output: 20
```

Slicing [:]
Extracts a portion of the array from start to end index.

```
arr = np.array([1, 2, 3, 4, 5])
print(arr[1:4])
# Output: [2 3 4]
```

dtype
Attribute to check the data type of the array elements.

```
arr = np.array([1, 2, 3])
print(arr.dtype)
# Output: int64
```

astype()
Converts the array to a different data type.

```
arr = np.array([1.1, 2.9])
print(arr.astype('i'))
# Output: [1 2]
```

copy()
Creates a completely new array; changes do not affect original.

```
arr = np.array([1, 2, 3])
x = arr.copy()
x[0] = 9
print(arr)
# Output: [1 2 3]
```

view()
Creates a view of the original array; changes affect original.

```
arr = np.array([1, 2, 3])
x = arr.view()
x[0] = 9
```

NumPy Tutorial Cheat Sheet

```
print(arr)
# Output: [9 2 3]
```

shape
Attribute that returns a tuple representing array dimensions.

```
arr = np.array([[1, 2], [3, 4]])
print(arr.shape)
# Output: (2, 2)
```

reshape()
Changes the shape of an array without changing its data.

```
arr = np.array([1, 2, 3, 4])
print(arr.reshape(2, 2))
# Output: [[1 2] [3 4]]
```

nditer()
Efficient multi-dimensional iterator object.

```
arr = np.array([[1, 2], [3, 4]])
for x in np.nditer(arr): pass
# Output: 1 2 3 4 (iterated)
```

concatenate()
Joins a sequence of arrays along an existing axis.

```
arr1 = np.array([1, 2])
arr2 = np.array([3, 4])
print(np.concatenate((arr1, arr2)))
# Output: [1 2 3 4]
```

stack()
Joins a sequence of arrays along a new axis.

```
print(np.stack((arr1, arr2), axis=1))
# Output: [[1 3] [2 4]]
```

array_split()
Splits an array into multiple sub-arrays.

```
arr = np.array([1, 2, 3, 4, 5, 6])
print(np.array_split(arr, 3))
# Output: [arr([1, 2]), arr([3, 4])...]
```

where()
Returns indices of elements that satisfy a condition.

```
arr = np.array([1, 2, 3, 4, 5])
print(np.where(arr == 4))
# Output: (array([3]),)
```

sort()

Returns a sorted copy of an array.

```
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
# Output: [0 1 2 3]
```

Boolean Indexing
Filtering elements based on a condition.

```
arr = np.array([41, 42, 43, 44])
print(arr[arr > 42])
# Output: [43 44]
```

II. NumPy Random

rand()
Generates random float numbers between 0 and 1.

```
from numpy import random
print(random.rand())
# Output: 0.1234...
```

randint()
Generates a random integer.

```
print(random.randint(100))
# Output: 42 (random 0-100)
```

shuffle()
Modifies a sequence in-place by shuffling its contents.

```
arr = np.array([1, 2, 3, 4, 5])
random.shuffle(arr)
print(arr)
# Output: [3 1 5 2 4]
```

permutation()
Randomly permutes a sequence or returns a permuted range.

```
print(random.permutation([1, 2, 3]))
# Output: [2 1 3]
```

normal()
Draws random samples from a normal (Gaussian) distribution.

```
x = random.normal(loc=1, scale=2, size=(2, 2))
print(x)
# Output: [[1.2 -0.5] [2.1 0.8]]
```

binomial()
Draws samples from a binomial distribution.

```
x = random.binomial(n=10, p=0.5, size=5)
```

```
print(x)
# Output: [5 4 6 5 5]
```

poisson()
Draws samples from a Poisson distribution.

```
x = random.poisson(lam=2, size=5)
print(x)
# Output: [2 1 3 2 0]
```

uniform()
Draws samples from a uniform distribution.

```
x = random.uniform(size=(2, 2))
print(x)
# Output: [[0.1 0.9] [0.4 0.2]]
```

III. NumPy ufunc

frompyfunc()
Converts a Python function to a NumPy ufunc.

```
def my_add(x, y): return x+y
my_ufunc = np.frompyfunc(my_add, 2, 1)
print(my_ufunc([1], [2]))
# Output: [3]
```

add()
Adds arguments element-wise.

```
print(np.add([10, 20], [30, 40]))
# Output: [40 60]
```

trunc()
Removes decimals, returning float closest to zero.

```
print(np.trunc([-3.1666, 3.6667]))
# Output: [-3.  3.]
```

log2()
Base-2 logarithm.

```
print(np.log2([1, 4, 8]))
# Output: [0. 2. 3.]
```

sum()
Sum of array elements over a given axis.

```
print(np.sum([[1, 2], [3, 4]]))
# Output: 10
```

prod()

NumPy Tutorial Cheat Sheet

Product of array elements over a given axis.

```
print(np.prod([1, 2, 3]))
# Output: 6
```

diff()
Calculates the discrete difference between elements.

```
arr = np.array([10, 15, 25])
print(np.diff(arr))
# Output: [5 10]
```

lcm()
Returns the Lowest Common Multiple.

```
print(np.lcm(4, 6))
# Output: 12
```

gcd()
Returns the Greatest Common Divisor.

```
print(np.gcd(6, 9))
# Output: 3
```

sin()
Trigonometric sine (input in radians).

```
print(np.sin(np.pi/2))
# Output: 1.0
```

unique()
Returns sorted unique elements of an array.

```
arr = np.array([1, 1, 2, 3])
print(np.unique(arr))
# Output: [1 2 3]
```